INTERIOR PENALTY FINITE ELEMENT APPROXIMATION OF NAVIER-STOKES EQUATIONS AND APPLICATION TO FREE SURFACE FLOWS

THÈSE N^{O} 3971 (2007)

PRÉSENTÉE LE 5 DÉCEMBRE 2007

À LA FACULTÉ DES SCIENCES DE BASE
CHAIRE DE MODÉLISATION ET CALCUL SCIENTIFIQUE
PROGRAMME DOCTORAL EN MATHÉMATIQUES

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Christoph WINKELMANN

Diplomierter in Rechnergestützten Wissenschaften ETH
de nationalité suisse et originaire de Siselen (BE)

acceptée sur proposition du jury:

Prof. S. Morgenthaler, président du jury
Prof. A. Quarteroni, directeur de thèse
Dr M. Picasso, rapporteur
Prof. A. Reusken, rapporteur
Prof. L. Tobiska, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Suisse
2007

Abstract

In the present work, we investigate mathematical and numerical aspects of interior penalty finite element methods for free surface flows. We consider the incompressible Navier-Stokes equations with variable density and viscosity, combined with a front capturing model using the level set method.
We formulate interior penalty finite element methods for both the Navier-Stokes equations and the level set advection equation. For the two-fluid Stokes equations, we propose and analyze an unfitted finite element scheme with interior penalty. Optimal a priori error estimates for the velocity and the pressure are proved in the energy norm.
A preconditioning strategy with adaptive reuse of incomplete factorizations as preconditioners for Krylov subspace methods is introduced and applied for solving the linear systems.
Different and complementary solutions for reducing the matrix assembly time and the memory consumption are proposed and tested, each of which is applicable in general in the context of either multiphase flow or interior penalty stabilization.
As level set reinitialization method, we apply a combination of the interface local projection and a fast marching scheme. We provide for the latter a reformulation of the distance computation algorithm on unstructured simplicial meshes in any spatial dimension, allowing for both an efficient implementation and geometric insight.
We present and discuss numerical solutions of reference problems for the one-fluid Navier-Stokes equations and for the level set advection problem. Solutions of benchmark problems in two and three dimensions involving one or two fluids are then approximated, and the results are compared to literature values. Finally, we describe software design techniques and abstractions for the efficient and general implementation of the applied methods.

Keywords: Navier-Stokes equations, free surface, finite elements, interior penalty stabilization, unfitted elements, front capturing, level set method, reinitialization, preconditioning

Version abrégée

Le sujet de cette thèse est l'étude d'aspects mathématiques et numériques de méthodes d'éléments finis avec pénalisation interne pour des écoulements à surface libre. Nous considérons les équations de Navier-Stokes incompressible à densité et viscosité variables, combinées avec un modèle de *front capturing* utilisant la méthode *level set*.
Nous formulons des méthodes d'éléments finis avec pénalisation interne pour les équations de Navier-Stokes et pour l'équation d'advection level-set. Pour les équations de Stokes bifluide, nous proposons et analysons un schéma d'éléments finis à maillage non conforme, muni d'une stabilisation par pénalisation interne. Des estimations d'erreur a priori optimales sont prouvées pour la vitesse et la pression en norme d'énergie.
Une stratégie de préconditionnement avec réutilisation adaptative de factorisations incomplètes comme préconditionneurs pour méthodes de sous-espaces de Krylov est introduite et appliquée pour résoudre les systèmes linéaires.
Des solutions différentes et complémentaires pour la réduction du temps de l'assemblage des matrices et de l'utilisation de mémoire sont proposées et testées. Chacune d'entre elles est applicable en général soit dans le contexte des écoulements multifluides, soit dans celui des méthodes avec stabilisation à pénalisation interne.
Comme méthode de réinitialisation level set, nous appliquons une combinaison de projection locale à l'interface avec un schéma de *fast marching*. Pour ce dernier, nous fournissons une reformulation de l'algorithme pour le calcul des distances, permettant à la fois une implémentation efficace et une interprétation géométrique intuitive.
Nous présentons et discutons des solutions numériques de problèmes de référence pour les équations de Navier-Stokes monofluide et pour le problème d'advection level set. Des solutions de problèmes de benchmark mono- et bifluides en deux et trois dimensions sont approximées ensuite, et les résultats obtenus sont comparés aux valeurs de la littérature. Finalement, nous décrivons des techniques de design de logiciel et des abstractions qui mènent à une implémentation efficace et générale des méthodes appliquées.

mots clés: équations de Navier-Stokes, surface libre, éléments finis, stabilisation à pénalisation interne, maillage non conforme, front capturing, méthode level set, réinitialisation, préconditionnement

Zusammenfassung

Die vorliegende Dissertation befasst sich mit mathematischen und numerischen Aspekten von Finite-Elemente-Methoden mit *Internal-Penalty*-Stabilisierung für Strömungen mit freien Oberflächen. Wir betrachten die inkompressiblen Navier-Stokes-Gleichungen mit variabler Dichte und Viskosität, kombiniert mit einem *Front-Capturing*-Modell unter Verwendung der *Level-Set*-Methode.
Wir formulieren Finite-Elemente-Methoden mit Internal-Penalty-Stabilisierung sowohl für die Navier-Stokes-Gleichungen als auch für die Transportgleichung der Level-Set-Funktion. Für die zweiphasigen Stokes-Gleichungen schlagen wir ein ebenso stabilisiertes Finite-Element-Schema auf nichtkonformem Gitter vor und beweisen optimale a-priori-Fehlerabschätzungen für Geschwindigkeit und Druck in der Energienorm.
Eine Vorkonditionierungsstrategie mit adaptiver Wiederverwendung von unvollständigen Faktorisierungen als Vorkonditionierer für Krylov-Unterraum-Methoden wird eingeführt und für die Lösung der linearen Systeme angewendet.
Verschiedene sich ergänzende Lösungen zur Reduktion der Assemblierungszeit der Matrizen und des Speicherbedarfs werden vorgestellt und getestet. Jeder dieser Ansätze ist allgemein verwendbar entweder im Kontext von Mehrphasenströmungen oder von Internal-Penalty-Stabilisierung.
Als Reinitialisierungsmethode der Level-Set-Funktion verwenden wir eine Kombination aus oberflächenlokaler Projektion und einem *Fast-Marching*-Schema. Für Letzteres stellen wir eine Neuformulierung des Distanzberechnungsalgorithmus auf unstrukturierten Simplexgittern in beliebiger Raumdimension vor, die einerseits eine effiziente Implementation erlaubt und andererseits eine intuitive geometrische Interpretation ermöglicht.
Wir stellen numerische Lösungen von Referenzproblemen für die einphasigen Navier-Stokes-Gleichungen und für das Level-Set-Transportproblem vor und diskutieren diese. Anschliessend werden Lösungen von zwei- und dreidimensionalen ein- und zweiphasigen Benchmarkproblemen angenähert und die Resultate werden mit Literaturwerten verglichen.
Schlussendlich beschreiben wir Software-Design-Techniken und Abstraktionen für eine effiziente und allgemeine Implementierung der angewendeten Methoden.

Schlagwörter: Navier-Stokes-Gleichungen, freie Oberfläche, finite Elemente, Interior-Penalty-Stabilisierung, nichtkonforme Gitter, Front-Capturing, Level-Set-Methode, Reinitialisierung, Vorkonditionierung

Riassunto

L'obiettivo di questo lavoro è lo studio di aspetti matematici e numerici del metodo degli elementi finiti con penalizzazione interna per correnti a superficie libera. Il problema in considerazione è dato dalle equazioni di Navier-Stokes per un fluido incomprimibile a densità e viscosità variabile, combinate con un modello di *front capturing* utilizzando il metodo *level set*.
Si considera il metodo degli elementi finiti con penalizzazione interna per le equazioni di Navier-Stokes e per l'equazione di trasporto della funzione level set. Per le equazioni di Stokes bifluido, si propone ed analizza uno schema di elementi finiti con griglia non conforme, stabilizzato mediante penalizzazione interna. Si dimostrano stime a priori ottimali dell'errore per la velocità e per la pressione nella norma energia.
Una strategia di precondizionamento basata sulla riutilizzazione adattativa di fattorizzazioni incomplete come precondizionatori per metodi di sottospazi di Krylov è introdotta e applicata per risolvere i sistemi lineari.
Inoltre si propongono e si applicano soluzioni varie e complementarie per ridurre il tempo di assemblaggio delle matrici e l'occupazione di memoria. Ognuna di queste soluzioni può essere applicata in generale oppure nel caso di correnti multifluido oppure nel caso di metodi con stabilizzazione a penalizzazione interna.
Per la reinizializzazione level set, si applica una combinazione di proiezione locale all'interfaccia con uno schema di *fast marching*. Per questo ultimo si presenta una nuova formulazione dell'algoritmo per il calcolo delle distanze, che permette un'implentazione efficace e un'interpretazione geometrica intuitiva.
Si presentano e si discutono soluzioni numeriche di problemi di referenza per le equazioni di Navier-Stokes monofluido e per il problema di trasporto level set. Delle soluzioni di problemi di benchmark mono- e bifluido in due e tre dimensioni sono approssimate poi, ed i risultati ottenuti sono paragonati ai valori della letteratura. Infine, si descrivano delle techiche di software design e delle astrazioni che permettono di implementare in modo efficace e generale i metodi applicati.

parole chiave: equazioni di Navier-Stokes, superficie libera, metodo degli elementi finiti, stabilizzazione con penalizzazione interna, griglia non conforme, front capturing, metodo level set, reinitializzazione, precondizionamento

Acknowledgements

First of all, I want to express my gratitude to Professor Alfio Quarteroni for giving me the opportunity to work on a fascinating subject in a stimulating research environment. His precious advice and the trust he has always placed in me have been fundamental for my scientific development during my years in his group.
I thank Professor Arnold Reusken, Professor Lutz Tobiska and Dr. Marco Picasso for having accepted to be members of the jury, and for reading this manuscript carefully. Professor Stephan Morgenthaler is also acknowledged for having accepted to preside the jury.
I am most grateful to Professor Erik Burman for the introduction to the wonderful world of stabilized finite elements and their analysis. I particularly appreciate his patient attitude, his uncountable and valuable recommendations and his availability for any kind of questions and for proof-reading this manuscript. Many thanks to Professor Christophe Prud'homme for having generously invested a big amount of work into the management of LifeV and the development of Life, and for proof-reading a part of this manuscript. This work has benefitted from the collaboration with Samuel Quinodoz, Dr. Daniele di Pietro, and Lorenza Santini. I am very grateful for their precious contributions. I would also like to thank Dr. Nicola Parolini and Shuren Hysing for sharing thoughts, ideas and opinions about the level set method and free surface flow simulation.
The financial support of the present work by the Swiss taxpayers via the Swiss National Science Foundation (projects 103809 and 112166) and the École Polytechnique Fédérale de Lausanne is gratefully acknowledged.
I also want to thank all present and former members of the Chair of Modelling and Scientific Copmuting CMCS, as well as all the members of the Institute of Analysis and Scientific Computing IACS for the many discussions and meals in a pleasant atmosphere. Special thanks to Benjamin Stamm for joining the Swiss german fraction of the group, to Dr. Simone Deparis for his support in turbulent times, and to Mme Jacqueline Mosetti for her unfailing assistance.
It is a pleasure to thank all the members of Lausanne Sports Aviron and of Lausanne university rowing, too numerous to be listed here, for all these wonderful moments shared on lakes and rivers throughout Europe. I am very much indebted to Adrian Burri for having been a fellow student in numerics, a team partner in rowing, but before all for having been and still being an invaluable friend.
Finally, my thanks go to my mother Renate, my father Ueli and my brother Johannes for the love, support, and understanding we share beyond all obstacles. Ich danke Euch aus tiefstem Herzen.

Contents

Introduction

Motivation and Objectives

Multiphase flow is an important subject in applied mathematics because it occurs in many situations under different forms. Wherever immiscible fluids touch each other, a free surface emerges. The fluids can form jets [69], bubbles [53],[112], droplets, waves [75] and films. These forms provide a large spectrum of applications: water waves on rivers, lakes and oceans [5],[89] interacting with vessels and shores [116],[62]; ship hydrodynamics [85]; injection, casting and extrusion of polymers and liquid metals [22]; insects walking on water surfaces [54]; but also bubble column chemical reactors and ink jet printers to name but a few. These applications give rise to a vast literature studying various aspects of the behaviour of multiphase flows.

Mathematical works concerning free boundary problems have begun around 1800 with Lagrange and, later, with the works of Stokes and Saint-Venant. The incompressible *Navier-Stokes equations* are by now widely accepted as mathematical model for incompressible flow of viscous fluids in general. In the case of several fluids, the equations are considered for each phase separately. The different phases are then coupled by suitable interface conditions describing also the effect of surface tension. The evolution of the domains occupied by the different fluids is part of the solution. Alternatively, and equivalently, multiphase flow can also be modelled considering the different phases as one inhomogeneous fluid, i.e., a fluid with properties like density and viscosity that are not constant in the whole domain. The evolution of the distribution of these properties is part of the solution. Effects of surface tension can be included in this perspective as forces localized on the interfaces.

These two paradigms are also encountered under the aspect of numerical modelling. The first approach gives rise to the so-called *front tracking methods*, whereas the second one leads to *front capturing methods*. In this work, we discuss front capturing methods for two-phase flow problems and consider a spatial domain with fixed boundaries occupied by a fluid with piecewise constant, but globally variable density and viscosity. We model the evolution of the interface by the *level set method*. In this method, the interface is represented as the zero level set of a continuous function ϕ defined on the entire domain, called level set function. The two subdomains occupied by one fluid each are then given by the set of points where the level set function is positive or negative, respectively, and the evolution of these domains and the interface is described by a transport equation of the level set function under the fluid velocity field. More details on the level set method can be found e.g. in the early paper by Dervieux and Thomasset [31] and the monographs [95] and [81].

To a given configuration of an interface shape and a fluid distribution, there are infinitely

many level set functions defining this configuration. Level set functions with gradients that are neither too steep nor too flat are preferable from both analytical and numerical viewpoints. Ideally, the level set function ϕ satisfies $|\nabla\phi| = 1$. Even if satisfied for the initial configuration, this property is not preserved under advection with the fluid velocity field. Therefore, it is necessary to restore this property from time to time [31]. This process is called *reinitialization*. One objective of this thesis is to find an accurate and efficient numerical reinitialization method suitable for two and three space dimensions.

Numerical methods are needed not only for the evolution of the interface, but also for the determination of the velocity and pressure characterizing the flow itself. The application of standard Galerkin finite element methods calls for different discrete finite element spaces for velocities and pressure (see [13]) in order to ensure stability. Many different stabilized methods have been proposed to allow to use the same spaces for pressure and velocity. One of these methods is the *interior penalty method* (see [18]). To investigate algorithmic aspects of this method when applied to the Navier-Stokes equations is another objective of the present work.

It is well known that discontinuous coefficients in elliptic problems lead to loss of accuracy and pollution effects perturbing the solution close to the discontinuity. The same can be expected in the context of the Navier-Stokes equations with discontinuous densities and viscosities. Similar problems are also caused by the presence of surface tension. The artificial flow fields induced are known as *spurious velocities* (see e.g. [40]). This problem shall also be adressed by this thesis.

Contributions of the Present Work

This thesis provides contributions to different areas of numerical analysis, scientific computing and software design.

A contribution to numerical analysis is given in Section 2.5, where we present an original analysis of an unfitted finite element scheme with interior penalty for the Stokes problem in two dimensions. The scheme solves the problem of discontinuous density and viscosity in two-fluid flow equations and can be generalized to the time dependent Navier-Stokes equations. We prove optimal energy norm estimates for the velocity and the pressure.

Chapter 3 contains several contributions to the field of scientific computing. A preconditioning strategy with reusable preconditioner for the solution of sequences of linear systems with evolving matrices is introduced in Subsection 3.2.3. Its conceptual simplicity based on few hypotheses makes it useful for applications that go far beyond the cases considered in this thesis.

An algorithmic aspect of interior penalty methods for flow equations is investigated for the first time in Subsection 3.3.2. We study a strategy for applying interior penalty methods without the drawback of a bigger matrix. The price to pay is an increase in the number of nonlinear iterations, a price which is however compensated by the reduced cost of these iterations due to the smaller matrix.

In Section 3.4, we present an idea to reduce substantially the cost of constructing the finite

element matrix of a two-fluid problem and demonstrate its efficiency. The idea, which seems specific to two-fluid flow problems at first, can be adapted to other cases, most interestingly also to problems with adaptive mesh refinement for any physical problem, not necessary flow related.

The problem of reinitialization is adressed in Section 3.5, where we combine the interface local projection method for optimal accuracy with the fast marching method for optimal efficiency. We introduce a new formulation of the core piece of the fast marching method, the distance computation, providing both geometric insight and an efficient implementation for arbitrary spatial dimension.

Chapter 6 finally contains some propositions of suitable abstractions at a software design level. The role of the Oseen problem as versatile framework for a wide range of applications is pointed out, and a unique combination of programming techniques is presented for keeping the cost of its generality affordable. A general and efficient Oseen solver can be used for stationary and time dependent problems, for the Darcy, Stokes, Euler and Navier-Stokes equations, for formulations with fixed and moving grids, and for constant coefficients as well as for variable ones.

Thesis Outline

In *Chapter 1*, we derive a mathematical model for the general case of a fluid with variable density and viscosity and thus also appropriate for modelling two fluid free surface flows. We also discuss possible types of boundary conditions which are useful in the context of two fluid flows. Moreover, modelling aspects related to the specific nature of free surface flows are adressed. Finally, we shortly recall known well-posedness results for the Navier-Stokes equations with variable coefficients with and without surface tension.

In *Chapter 2*, we introduce different finite element methods for incompressible flow in general, and for the Navier-Stokes equations with discontinuous density and viscosity in particular. We present an interior penalty stabilized finite element method for the Oseen equation with equal order interpolation. This method is well suited for incompressible flow computations, specially for two fluid flows without surface tension. We recall the most important theoretical results known for continuous interior penalty methods for incompressible flows with constant density and viscosity. We also give a short overview of known results for finite element methods applied to flows with discontinuous density and viscosity. We then shortly present the well-known Taylor-Hood finite element scheme which will serve both for comparison in general, and as an alternative in the case of two fluid flows with surface tension. A first step towards an analysis of an interior penalty finite element scheme for two fluid flow is presented, where we consider the two dimensional stationary Stokes equations. Finally, an interior penalty stabilized finite element formulation for the advection-reaction equation is presented. This scheme will be used for the transport equation of the interface in two fluid flow computations.

In *Chapter 3*, we adress some algorithmic aspects of incompressible flow simulations with continuous interior penalty finite element methods in general, and of two fluid flow computations in particular. We present the iterative schemes we use to treat time dependence, coupling,

and nonlinearity of the flow equations. Then, we give a short overview of linear algebraic solution approaches and present a general strategy for the adaptive reuse of preconditioners. Furthermore, we suggest solutions to reduce the two main drawbacks of continuous interior penalty methods: its costly assembly and its extended stencil. In addition, a simple but effective strategy of reducing the cost of the matrix assembly for two fluid flows is presented. Finally, we recall different level set reinitialization methods and suggest a combination of the interface local projection and a fast marching method in a new formulation.

In *Chapter 4*, we consider different problems with nontrivial known exact solutions. The aim is to test the convergence behaviour of the finite element methods introduced in Chapter 2 combined with the iterative schemes from Chapter 3 in simple cases where all kind of errors can be computed because the exact solution is known. We consider stationary and time dependent solutions of the Navier-Stokes equations in two and three space dimensions, plus a benchmark problem for the advection equation oriented towards interface evolution modelling.

In *Chapter 5*, we consider several benchmark flow problems where no analytic solution is known. The aim here is to compare our results to literature values in order to compare our methods to well established schemes. We first consider a benchmark problem for the one-fluid Navier-Stokes equations in two and three dimensions. Then, we solve a benchmark problem for the two-fluid Navier-Stokes equations in two dimensions. Finally, we present some computations of free-surface flow with topology changes of the interface, in order to test the ability of the method to cope with this kind of phenomena.

In *Chapter 6*, we shortly present the software environment *Life*, in which the methods and algorithms from Chapters 2 and 3 were implemented and with which the results in Chapters 4 and 5 were obtained. We first give an overview of the project development and some details about the new mathematical kernel. Then we discuss the contributions added to the project in the framework of the present thesis.

Chapter 1

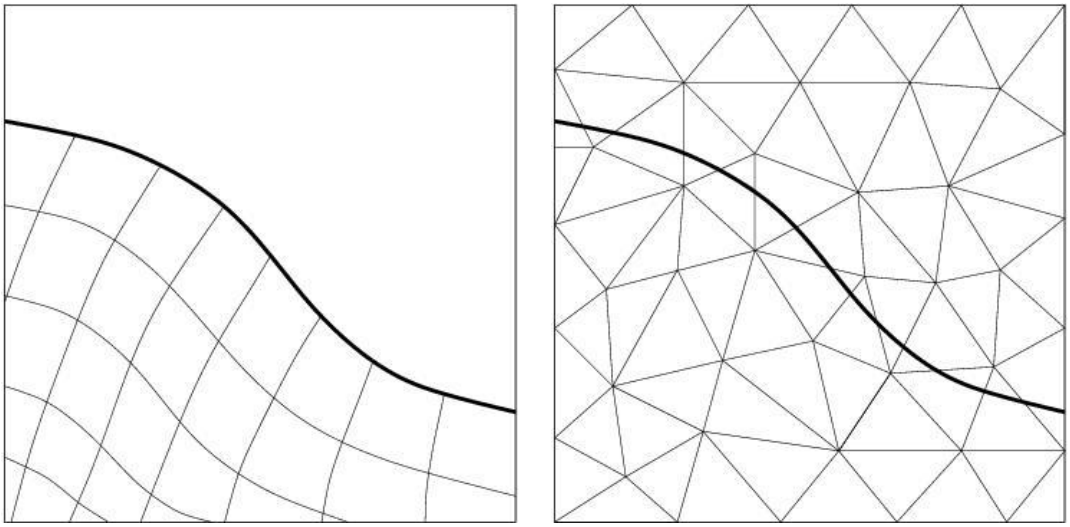
Mathematical Model

Free surface flows occur in many situations under different forms. Depending on the time and length scales as well as the physical properties of the substances involved, many phenomena may or may not have an essential influence on the behaviour of the fluids to be studied. The fluids can interact with solid structures, deforming them and being affected by them. Heat transfer can change the temperature and properties dependent on temperature like the density or the viscosity. It may even make liquids solidify or solids melt. The surface tension of the surface separating two fluids influences in its turn the characteristic size of its structure. Viscous effects will damp turbulence, and their nonlinearity may cause effects that seem often counter-intuitive. Low viscosity on the other hand will cause the flow to develop vortices and other turbulent structures, depending on geometry and velocity. The compressibility of the fluids investigated is often but not always neglectable. Also, some fluids may react chemically with other fluids or also solid structures.
In the present work, we will focus on laminar flows of viscous but Newtonean fluids subject to surface tension. The fluid can therefore be modeled by the Navier-Stokes equations. We will neglect the other aspects mentioned above in order to focus on the specificities of the presence of the free surface.
For modelling such flows, two families of models are generally used:

- *Front tracking methods*, see Figure 1.1(a), consider the free surface as the boundary of a moving domain, on which special free surface boundary conditions are specified. Inside the domain, a standard flow model is applied, taking into account the fact that the domain is not fixed but moving. The fluid on the other side of the boundary is usually neglected or its influence is modelled with some simplified approach, but it is not usually simulated on its own (see e.g. [77]).

- *Front capturing methods*, see Figure 1.1(b), consider two fluids in a spatial domain with fixed boundaries. The two fluids are separated by a free surface. These two fluids can equivalently be seen as one fluid with piecewise constant, but globally variable properties like density and viscosity. A nonstandard flow model is needed to account for this variability, whereas no special solutions are needed for the moving domain and for the matching conditions at the free surface (see e.g. [47],[52]).

A review of state of the art models and methods for solving free surface problems can be found in [23]
In the present work, we apply front capturing methods, considering the two fluids as one fluid



(a) front tracking (b) front capturing

Figure 1.1: Typical grid topologies in two dimensions for front tracking (a) and front capturing (b) free surface methods. The thick line represents the free surface.

with piecewise constant, but globally variable density and viscosity. More precisely, in this chapter we derive a mathematical model for the general case of a fluid with variable density and viscosity and thus also appropriate for modelling two fluid free surface flows. Moreover, modelling aspects related to the specific nature of free surface flows are discussed.

1.1 Flow Model

1.1.1 Navier-Stokes Equations with Variable Density and Viscosity

We denote time by t and the Cartesian spatial coordinates by $\boldsymbol{x} = \{x_i\}_{i=1}^d$, $d \in \{2, 3\}$. The vectorial operator of spatial derivatives is denoted by $\boldsymbol{\nabla} = \{\partial_{x_i}\}_{i=1}^d$, where ∂_ξ is the partial derivative with respect to ξ. For the sake of generality, we consider a viscous incompressible fluid, whose density ρ and dynamic viscosity μ depend on space and time. Here we do not consider temperature dependence, but the presented method may be applied to more complex models that include heat transfer. Within a fixed spatial domain $\Omega \subset \mathbb{R}^d$, and during the time interval $(0, T)$, i.e., for $(\boldsymbol{x}, t) \in \Omega \times (0, T)$, the evolution of the velocity $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t)$ and the pressure $p = p(\boldsymbol{x}, t)$ of the fluid is modeled by the incompressible Navier-Stokes equations:

$$\rho \partial_t \boldsymbol{u} + \rho(\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{u} - \boldsymbol{\nabla} \cdot (2\mu \mathbf{D}(\boldsymbol{u})) + \boldsymbol{\nabla} p = \boldsymbol{f} \quad \text{in } \Omega \times (0, T), \tag{1.1}$$

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \quad \text{in } \Omega \times (0, T), \tag{1.2}$$

where $(\mathbf{D}(\boldsymbol{v}))_{ij} := \frac{1}{2}\left(\partial_{x_j}(\boldsymbol{v})_i + \partial_{x_i}(\boldsymbol{v})_j\right)$, $i, j = 1 \ldots d$, is the *symmetric gradient* of $\boldsymbol{v}$, also called *deformation rate tensor* or *strain rate tensor*, and $\boldsymbol{f}$ denotes a volumic force, e.g. gravitation.

Equation (1.1) enforces the conservation of momentum, whereas equation (1.2) expresses the incompressibility constraint, equivalent to the conservation of volume. Note that incompressibility is not in contradiction with variable density. Incompressibility means that one fluid

parcel does not change volume and thus density, whereas variable density means that different fluid parcels may have different densities.
The last two terms of the left hand side of equation (1.1) can also be written as $-\boldsymbol{\nabla}\cdot\mathbf{T}(\boldsymbol{u},p)$, where

$$\mathbf{T}(\boldsymbol{u},p)=2\mu\mathbf{D}(\boldsymbol{u})-\mathbf{I}p$$

is the *stress tensor* and $\mathbf{I}$ is the $d\times d$ identity tensor. The divergence of a tensor $\mathbf{S}$ of rank two is the d-vector with components

$$(\boldsymbol{\nabla}\cdot\mathbf{S})_i=\sum_{k=1}^{d}\partial_{x_k}\mathbf{S}_{ik}.$$

For a detailed derivation, motivation and justification of this model, see [67].
The density ρ and the viscosity μ may vary with space and time. A separate model for their evolution is needed. Usually, a mass balance leads to the equation

$$\begin{aligned}\mathrm{d}_t\rho=\partial_t\rho+\boldsymbol{u}\cdot\boldsymbol{\nabla}\rho=0 \quad &\text{in } \Omega\times(0,T),\\ \rho|_{t=0}=\rho_0 \quad &\text{in } \Omega\end{aligned}\tag{1.3}$$

In cases where the viscosity μ can be expressed as a function of the density $\mu(\rho)$, this relationship together with (1.3) constitute a model for the evolution of ρ and μ.
Suitable models for the special case of two fluid flow are described in Subsection 1.1.3.
The flow model has also to be completed with suitable initial and boundary conditions. For the initial conditions, it is clear that an initial velocity field $\boldsymbol{u}_0$ is necessary and sufficient:

$$\boldsymbol{u}|_{t=0}=\boldsymbol{u}_0 \quad \text{in } \Omega.$$

For the boundary conditions, the different possibilities are presented in Subsection 1.1.2.

1.1.2 Boundary Conditions

We partition the boundary $\partial\Omega$ of the domain Ω into a finite number n of subsets γ_i, $i=1\ldots n$. In order for the Navier-Stokes problem to be well-posed, suitable boundary conditions need to be specified on each subset γ_i. Many different types of boundary conditions are possible in principle. We refer to [91] and references therein for an extensive enumeration of these possibilities, and consider just the following ones:

Dirichlet Boundary Conditions

Dirichlet boundary conditions prescribe a velocity field $\boldsymbol{g}_D$:

$$\boldsymbol{u}=\boldsymbol{g}_D \quad \text{on } \gamma_i.\tag{1.4}$$

They are usually applied to impose an inflow velocity profile $\boldsymbol{g}_D$, or to model a wall moving with velocity $\boldsymbol{g}_D$. In the latter case, they are also called *no-slip boundary conditions*, as they impose the fluid not to slip but to stick at the wall.
Note that when Dirichlet boundary conditions are specified on the entire boundary $\partial\Omega$, the pressure is not uniquely defined. In this case, if $(\boldsymbol{u},p)$ is a solution of (1.1), (1.2) and (1.4),

then $(\boldsymbol{u}, p + c)$, $c \in \mathbb{R}$ is also a solution of the same set of equations. Instead, by partial integration of equation (1.2), $\boldsymbol{g}_D$ then has to satisfy the compatibility condition

$$\int_{\partial\Omega} \boldsymbol{g}_D \cdot \boldsymbol{n} \, \mathrm{d}s = 0,$$

otherwise the problem does not have any solution at all.

Neumann Boundary Conditions

Neumann boundary conditions prescribe a force $\boldsymbol{g}_N$ per unit area as the normal component of the stress tensor:

$$\mathbf{T}(\boldsymbol{u}, p)\boldsymbol{n} = 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n} = \boldsymbol{g}_N \quad \text{on } \gamma_i, \tag{1.5}$$

where $\boldsymbol{n}$ is the outer unit normal on γ_i. Neumann boundary conditions are used to model a given force per unit area $\boldsymbol{g}_N$ on the boundary, often with $\boldsymbol{g}_N = \mathbf{0}$ for what is called a *free outflow*. For vanishing velocity gradients, the force $\boldsymbol{g}_N$ corresponds to the pressure on the boundary. See also [51] for more details about the interpretation and implications of this type of boundary conditions.

Mixed Boundary Conditions

Mixed boundary conditions combine Dirichlet boundary conditions in the normal direction $\boldsymbol{n}$ with Neumann boundary conditions in the tangential direction(s) $\boldsymbol{\tau}$:

$$\begin{aligned} \boldsymbol{u} \cdot \boldsymbol{n} &= \boldsymbol{g}_D \cdot \boldsymbol{n} && \text{on } \gamma_i, \\ (\mathbf{T}(\boldsymbol{u}, p)\boldsymbol{n}) \cdot \boldsymbol{\tau} &= (2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}) \cdot \boldsymbol{\tau} = 0 && \text{on } \gamma_i, \qquad \forall \boldsymbol{\tau} : \boldsymbol{\tau} \cdot \boldsymbol{n} = 0. \end{aligned}$$

The choice $\boldsymbol{g}_D = \mathbf{0}$ models symmetry of the solution along γ_i, but also free slip on γ_i without penetration. In this case we talk about *free slip boundary conditions*.

Mixed Robin Boundary Conditions

In some situations, a smooth transition from slip to no-slip boundary conditions is desired. This can be realized by imposing Dirichlet boundary conditions in the normal direction, as for the free slip boundary conditions, and to replace the boundary condition in the tangential direction by Robin boundary conditions, a linear combination of Dirichlet and Neumann boundary conditions:

$$\boldsymbol{u} \cdot \boldsymbol{n} = \boldsymbol{g}_D \cdot \boldsymbol{n} \quad \text{on } \gamma_i, \tag{1.6}$$

$$\begin{aligned} &(\omega C_\tau \boldsymbol{u} + (1-\omega)(\mathbf{T}(\boldsymbol{u}, p)\boldsymbol{n})) \cdot \boldsymbol{\tau} = \\ &(\omega C_\tau \boldsymbol{u} + (1-\omega)(2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n})) \cdot \boldsymbol{\tau} = \omega C_\tau \boldsymbol{g}_D \cdot \boldsymbol{\tau} \quad \text{on } \gamma_i, \qquad \forall \boldsymbol{\tau} : \boldsymbol{\tau} \cdot \boldsymbol{n} = 0. \end{aligned} \tag{1.7}$$

Here, $\omega \in [0, 1]$ determines the regime. For $\omega = 0$, we have free slip boundary conditions, whereas for $\omega = 1$, we have no-slip boundary conditions. In practice, ω can be a smooth function of space and time, with values in $[0, 1]$, allowing thus a smooth transition between the two cases. This holds for $\boldsymbol{g}_D = \mathbf{0}$, but transition boundary conditions cover also the general Dirichlet case for $\boldsymbol{g}_D \neq \mathbf{0}$ and $\omega = 1$. The weight C_τ can be seen as conversion factor between velocities and force per unit area. This type of boundary conditions has been studied in more detail in [58].

1.1.3 Application to Free Surface Flow

A free surface flow can be modeled with the flow model (1.1)-(1.2). In this perspective, the free surface is an interface denoted by $\Gamma(t)$, cutting the domain Ω into two open subdomains $\Omega^+(t)$ and $\Omega^-(t)$. The initial position of the interface is known, $\Gamma(0) = \Gamma_0$, and the interface moves with the fluid velocity $\boldsymbol{u}$. On each subdomain, we have the constant densities and viscosities denoted by ρ^+, ρ^-, μ^+ and μ^-. We require $\rho^\pm > 0$ and $\mu^\pm > 0$.
Density and viscosity are then globally defined as follows:

$$\rho(\boldsymbol{x},t) = \begin{cases} \rho^- & \boldsymbol{x} \in \Omega^-(t) \\ \rho^+ & \boldsymbol{x} \in \Omega^+(t) \end{cases},$$
$$\mu(\boldsymbol{x},t) = \begin{cases} \mu^- & \boldsymbol{x} \in \Omega^-(t) \\ \mu^+ & \boldsymbol{x} \in \Omega^+(t) \end{cases}.$$

In order to model buoyancy effects, the gravitation force has to be introduced into the right hand side. It takes the form $\boldsymbol{f} = \rho \boldsymbol{g}$, where $\boldsymbol{g}$ is the vector of gravity acceleration.
As the viscosity is discontinuous across the interface, equation (1.1) can hold strongly only in $\Omega^+ \cup \Omega^-$. The two subdomains must then be coupled with suitable interface conditions (see e.g. [98]).
We denote by $\boldsymbol{n}_\Gamma$ the interface unit normal pointing from Ω^- into Ω^+ and by κ the interface curvature, defined as

$$\kappa = \sum_{i=1}^{d-1} \frac{1}{R_{\boldsymbol{\tau}_i}}, \tag{1.8}$$

where $R_{\boldsymbol{\tau}_i}$ are the radii of curvature along the principal vectors $\boldsymbol{\tau}_i$ which span the tangential space to the interface Γ. The sign of $R_{\boldsymbol{\tau}_i}$ is such that $R_{\boldsymbol{\tau}_i}\boldsymbol{n}_\Gamma$ points from Γ to the center of the circle approximating Γ locally.
The jump of a quantity v across the interface is denoted by $[\![v]\!]_\Gamma$ and defined as

$$\begin{aligned} [\![v]\!]_\Gamma(\boldsymbol{x},t) &= \lim_{\varepsilon \to 0^+} \left(v(\boldsymbol{x} + \varepsilon \boldsymbol{n}_\Gamma, t) - v(\boldsymbol{x} - \varepsilon \boldsymbol{n}_\Gamma, t)\right) \\ &= v|_{\Omega^+(t)}(\boldsymbol{x},t) - v|_{\Omega^-(t)}(\boldsymbol{x},t) \qquad \forall \boldsymbol{x} \in \Gamma(t). \end{aligned}$$

The interface conditions then read

$$[\![\boldsymbol{u}]\!]_\Gamma = \boldsymbol{0}, \tag{1.9}$$
$$[\![\mathbf{T}(\boldsymbol{u},p)\boldsymbol{n}_\Gamma]\!]_\Gamma = [\![2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}_\Gamma - p\,\boldsymbol{n}_\Gamma]\!]_\Gamma = -\sigma\kappa\boldsymbol{n}_\Gamma. \tag{1.10}$$

Equation (1.9) is called the *kinematic interface condition*. It expresses that all components of the velocity are continuous. In fact the normal component has to be continuous because there is no flow through the interface, whereas the tangential component(s) have to be continuous because both fluids are assumed viscous ($\mu^+ > 0$ and $\mu^- > 0$).
Equation (1.10) is refered to as the *dynamic interface condition*. It expresses that the normal stress jumps by the amount of the surface tension force. This force is proportional to the interface curvature and pointing to the direction of the interface normal. The surface tension coefficient σ depends on the fluid pairing, and in general also on temperature. We will assume it to be constant as all heat transfer effects are neglected.
Note that the evolution of the interface has to be compatible with the mass conservation equation (1.3). Mathematically, this equation has to be understood in the weak sense, i.e, in

the sense of distributions, as the density is discontinuous across the interface and its derivatives can by consequence only be interpreted weakly. Together with equations (1.1) and (1.2) it constitutes the model describing two fluid flow physically.
As this form of the mass conservation equation is often not convenient for numerical simulations, other equivalent models that describe the evolution of the interface $\Gamma(t)$ have been introduced. An overview of them is presented in Section 1.2.

1.2 Interface Evolution Modelling

We give here a short overview of different approaches for modelling the evolution of an interface $\Gamma(t)$ in a fixed domain Ω, and discuss the model chosen in the present work in more detail.

1.2.1 Explicit Interface Descriptions

An interface can be represented explicitly by a set of marker points or line segments (in 2D, surface segments in 3D) on the interface and transported by the fluid velocity $\boldsymbol{u}$.
In the case of marker points, introduced in [47], the connectivity of the interface between the points is not known and has to be reconstructed whenever needed. In order to simplify this task, additional markers are usually placed near the interface, marking Ω^+ or Ω^-. The advection of the markers is simple, and connectivity can change easily. However it is still somewhat cumbersome to reconstruct the interface from the marker distribution. Typically, it is also necessary to redistribute the markers, to introduce new ones or to discard existing ones.
Several markers can be connected to define a line or surface, either straight (plane) or curved, e.g. by nurbs. A set of such geometrical objects can now define the surface. Its evolution is modelled by the evolution of the constituting objects, and thus by the markers defining them. The connectivity of the interface is thereby conserved, which solves the difficulty of pure marker methods, and brings a new drawback in turn: Topological changes of the interface are allowed by the underlying physics but not by this description. Sophisticated procedures have to be applied to detect and handle interface breakup correctly.

1.2.2 Implicit Interface Descriptions

In front capturing methods, the interface is represented implicitly by the value of a scalar function $\phi : \Omega \times (0,T) \to I\!R$ that encodes at each point $\boldsymbol{x}$ to which subset it belongs: $\Omega^+(t)$ or $\Omega^-(t)$. A transport equation solved for ϕ then describes the evolution of the interface. By this feature, all implicit interface models share the advantage that topology changes of the interface are possible naturally in the model, and that these happen without special intervention.

Volume of Fluid Methods

The volume of fluid methods (VOF) were originally introduced by Hirt and Nichols [52]. Here ϕ is a piecewise constant function:

$$\phi(\boldsymbol{x},t) = \begin{cases} 1 & \boldsymbol{x} \in \Omega^+(t) \\ 0 & \boldsymbol{x} \in \Omega^-(t) \end{cases}$$

and the interface $\Gamma(t)$ is thus located at the discontinuity of the function ϕ.
Density and viscosity are then simply defined as

$$\begin{aligned} \rho &= \rho^- + (\rho^+ - \rho^-)\phi, \\ \mu &= \mu^- + (\mu^+ - \mu^-)\phi. \end{aligned} \tag{1.11}$$

The transport equation is usually discretized with finite volume methods, approximating ϕ by a constant value in each grid cell. Due to discretization errors and diffusive transport schemes, the approximation ϕ will take values between 0 and 1, which by the virtue of equation (1.11) can be (and usually are) interpreted as the volume fraction of the fluid occupying Ω^+. This explains the name *volume of fluid*. Volume fractions between 0 and 1 actually represent a mixture of the two fluids. As the fluids are assumed immiscible, this behaviour is not desired, specially because mixing effects may not stay concentrated near the interface but spread over the whole domain Ω. Like this, the supposedly sharp interface becomes more and more diffuse. Several techniques exist to limit this problem. Elaborate procedures have been developed for the reconstruction of normals and curvature of a diffuse interface. Volume of fluid methods have the advantage that applying a conservative discretization of the transport equation ensures mass conservation of the fluid, because the relation (1.11) between ϕ and ρ is linear.

Level Set Methods

In order to circumvent the problems with volume of fluid methods, Dervieux and Thomasset [31] proposed in 1980 to define the interface as the zero level set of a continuous *pseudo-density* function and to apply this method to flow problems. Their approach was then studied more systematically in [82] and subsequent publications, where the term *level set method* was coined. Their first application to flow problems was by Mulder, Osher and Sethian in 1992 [76]. In constrast with volume of fluid approaches, these methods allow to keep the interface sharp, as ϕ is defined as a *continuous* function such that

$$\begin{aligned} \phi(\boldsymbol{x}, t) &> 0 \quad \forall \boldsymbol{x} \in \Omega^+(t), \\ \phi(\boldsymbol{x}, t) &< 0 \quad \forall \boldsymbol{x} \in \Omega^-(t), \\ \phi(\boldsymbol{x}, t) &= 0 \quad \forall \boldsymbol{x} \in \Gamma(t). \end{aligned}$$

The function ϕ is called *level set function*, because the interface $\Gamma(t)$ is its zero level set, its isoline or isosurface associated to the value zero:

$$\Gamma(t) = \{\boldsymbol{x} \in \Omega : \ \phi(\boldsymbol{x}, t) = 0\}. \tag{1.12}$$

The density and the viscosity can now be expressed in function of ϕ as

$$\rho = \rho^- + (\rho^+ - \rho^-)H(\phi), \tag{1.13}$$
$$\mu = \mu^- + (\mu^+ - \mu^-)H(\phi), \tag{1.14}$$

where $H(\cdot)$ is the Heaviside function

$$H(\xi) = \begin{cases} 0 & \xi < 0 \\ 1 & \xi > 0 \end{cases}.$$

By construction, the interface stays sharp in a level set model, and the immiscible fluids do not start to mix. Also, the determination of the normals and the curvature of the interface are more straightforward and very natural. In turn, as the relation (1.13) is not linear, applying a conservative discretization of the transport equation for ϕ does not ensure mass conservation of the fluid after discretization. This is not a big problem however, as the mass error still disappears with grid refinement and is outweighed by advantages of the level set formulation.

1.2.3 The Level Set Interface Evolution Model

In the present work, we choose to model and to represent the interface with the level set approach introduced above and presented in detail in this section.

Interface Evolution

We describe the evolution of the free surface by an advection equation for the level set function:

$$\begin{aligned} \partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi = 0 \quad & \text{in } \Omega \times (0,T), \\ \phi = \phi_0 \quad & \text{in } \Omega \text{ at } t = 0, \\ \phi = \phi_{in} \quad & \text{on } \Sigma_{in}, \end{aligned} \tag{1.15}$$

where Σ_{in} is the inflow boundary

$$\Sigma_{in} = \{(\boldsymbol{x}, t) \in \partial\Omega \times (0,T) : \boldsymbol{u}(\boldsymbol{x}, t) \cdot \boldsymbol{n} < 0\}.$$

The flow equations (1.1)-(1.2) and the level set equation (1.15) are therefore coupled. Equation (1.15) can be derived as follows [76]: Let $\bar{\boldsymbol{x}}(t)$ be the path of a point on the interface $\Gamma(t)$. This point moves with the fluid, thus $\mathrm{d}_t \bar{\boldsymbol{x}}(t) = \boldsymbol{u}(\bar{\boldsymbol{x}}(t), t)$. Since the function ϕ is always zero on the moving interface, we must have

$$\phi(\bar{\boldsymbol{x}}(t), t) = 0.$$

Deriving with respect to time and applying the chain rule, we obtain

$$\partial_t \phi + \boldsymbol{\nabla} \phi \cdot \boldsymbol{u} = 0 \quad \text{on } \Gamma(t) \quad \forall t \in (0,T). \tag{1.16}$$

If we consider instead a path of a point in $\Omega^{\pm}$, we may require $\phi(\bar{\boldsymbol{x}}(t), t) = \pm c$, $c > 0$, in order to ensure that the sign of $\phi(\bar{\boldsymbol{x}}, t)$ does not change and that $\bar{\boldsymbol{x}}(t) \in \Omega^{\pm}(t)$ $\forall t$ thereby. In this way, equation (1.16) generalizes to the whole domain Ω, which gives us equation (1.15).
We can now verify that mass conservation is satisfied: Using (1.13), we obtain formally:

$$\begin{aligned} \partial_t \rho + \boldsymbol{u} \cdot \boldsymbol{\nabla} \rho &= (\rho^+ - \rho^-)(\partial_t H(\phi) + \boldsymbol{u} \cdot \boldsymbol{\nabla} H(\phi)) \\ &= (\rho^+ - \rho^-)\delta(\phi)(\partial_t \phi + \boldsymbol{u} \cdot \boldsymbol{\nabla} \phi) \end{aligned} \tag{1.17}$$

where $\delta(\cdot)$ denotes the Dirac delta function. By equation (1.15), the third factor in (1.17) is zero. Hence equation (1.3) holds and the mass conservation is satisfied by the level set interface evolution model.

Interface Related Quantities

In the context of two fluid flow, the interface normal and curvature are of particular interest. Namely the surface tension is proportional to the curvature and acting in the normal direction. We give here an intuitive derivation of these quantities in function of ϕ, without going into the details of differential geometry. See e.g. [99] for a detailed and rigorous derivation.
The unit normal $\boldsymbol{n}_\Gamma$ is orthogonal to all tangential directions $\boldsymbol{\tau}$, which in turn are characterized by the fact that the directional derivative of ϕ in any tangential direction must vanish:

$$0 = \partial_{\boldsymbol{\tau}}\phi = \boldsymbol{\nabla}\phi \cdot \boldsymbol{\tau} \quad \text{on } \Gamma.$$

The gradient of ϕ is thus orthogonal to all tangential directions, and we can define the interface unit normal by normalizing it:

$$\boldsymbol{n}_\Gamma = \frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}. \tag{1.18}$$

Note that by this definition, $\boldsymbol{n}_\Gamma$ points from Ω^- into Ω^+. Moreover, as ϕ is defined not only on the interface but in the whole domain, the expression for the normal generalizes naturally to the entire domain, too.
In order to derive the expression for the curvature, we need to consider the principal tangential direction(s) $\boldsymbol{\tau}_i$, $i = 1 \dots d-1$. These are the directions in which the interface is approximated by a circle (cylinder), i.e., the directional derivative of $\boldsymbol{n}_\Gamma$ in direction $\boldsymbol{\tau}_i$ has itself direction $\boldsymbol{\tau}_i$:

$$\partial_{\boldsymbol{\tau}_i}\boldsymbol{n}_\Gamma = \boldsymbol{\nabla}\boldsymbol{n}_\Gamma\,\boldsymbol{\tau}_i = -\kappa_i\boldsymbol{\tau}_i, \quad \kappa_i \in \mathbb{R}, \quad i = 1 \dots d-1 \tag{1.19}$$

The bigger $|\kappa_i|$, the more curved is the surface in this direction, and the κ_i are in fact called *principal curvatures*. It follows from straightforward computations that $\kappa_i = (R_{\boldsymbol{\tau}_i})^{-1}$, where the values $R_{\boldsymbol{\tau}_i}$ are the radii of the approximating circles (cylinders) as in equation (1.8).
We can see from equation (1.19) that the $d-1$ values $-\kappa_i$ are eigenvalues of the $d \times d$-tensor $\boldsymbol{\nabla}\boldsymbol{n}_\Gamma$. By (1.18), $\boldsymbol{n}_\Gamma$ is (essentially) a gradient field which is smooth near the interface. The rank two tensor $\boldsymbol{\nabla}\boldsymbol{n}_\Gamma$ is thus (essentially) a tensor of second derivatives of a smooth function and thereby symmetric. So it has one more real eigenvalue, whose associated eigenvector must be $\boldsymbol{n}_\Gamma$, because the eigenvectors of a symmetric tensor are orthogonal. It is easy to see that the respective eigenvalue is zero:

$$(\boldsymbol{\nabla}\boldsymbol{n}_\Gamma\,\boldsymbol{n}_\Gamma)_i = \sum_{j=1}^{d}(\partial_{x_i}n_j)n_j = \sum_{j=1}^{d}\frac{1}{2}\partial_{x_i}(n_j^2) = \frac{1}{2}\partial_{x_i}\,|\boldsymbol{n}_\Gamma|^2 = 0,$$

as $|\boldsymbol{n}_\Gamma| = 1$ by construction (1.18).
Starting from equation (1.8), we obtain for the curvature

$$\kappa = \sum_{i=1}^{d-1}\frac{1}{R_{\boldsymbol{\tau}_i}} = \sum_{i=1}^{d-1}\kappa_i = -\mathrm{tr}(\boldsymbol{\nabla}\boldsymbol{n}_\Gamma) = -\boldsymbol{\nabla}\cdot\boldsymbol{n}_\Gamma,$$

and using equation (1.18), we get

$$\kappa = -\boldsymbol{\nabla}\cdot\left(\frac{\boldsymbol{\nabla}\phi}{|\boldsymbol{\nabla}\phi|}\right). \tag{1.20}$$

Initial Condition

We know the position of the interface at $t = 0$, Γ_0, nevertheless the associated level set function ϕ_0 is not uniquely defined. The freedom of choice can be used to simplify further subsequent tasks. We notice that steep gradients of ϕ make the numerical solution of equation (1.15) more difficult (see e.g. [91]), whereas flat gradients decrease the numerical stability when determining Γ from ϕ. A good compromise is thus the further constraint $|\boldsymbol{\nabla}\phi| = 1$.
A function which fulfills this constraint is the distance function

$$\mathrm{dist}(\boldsymbol{x};\Gamma) = \min_{\boldsymbol{y}\in\Gamma} |\boldsymbol{x} - \boldsymbol{y}|\,,$$

which at each point $\boldsymbol{x}$ takes the value of the closest Euclidean distance from $\boldsymbol{x}$ to Γ. Multiplying this function by -1 on Ω^-, we obtain the *signed distance function*:

$$\mathrm{sdist}(\boldsymbol{x};\Gamma) = \begin{cases} \mathrm{dist}(\boldsymbol{x};\Gamma) & \boldsymbol{x}\in\Omega^+ \\ 0 & \boldsymbol{x}\in\Gamma \\ -\mathrm{dist}(\boldsymbol{x};\Gamma) & \boldsymbol{x}\in\Omega^- \end{cases}.$$

It is thus usual and reasonable to choose ϕ_0 representing an initial interface Γ_0 as $\phi_0(\boldsymbol{x}) = \mathrm{sdist}(\boldsymbol{x};\Gamma_0)$.
It is interesting to note that if $|\boldsymbol{\nabla}\phi| = 1$, the expressions the interface normal and curvature simplify further:

$$\boldsymbol{n}_\Gamma = \boldsymbol{\nabla}\phi \qquad \text{and} \qquad \kappa = -\boldsymbol{\nabla}\cdot\boldsymbol{\nabla}\phi = -\Delta\phi.$$

Reinitialization

Unfortunately, the property $|\boldsymbol{\nabla}\phi| = 1$ is not preserved under advection of ϕ with the fluid velocity $\boldsymbol{u}$. This is not a problem as long as $|\boldsymbol{\nabla}\phi|$ does not stay too far from 1, which however cannot be guaranteed in general. Two different strategies can be followed to cope with this issue.
One approach is to determine an advection velocity field that gives the same interface motion as the fluid velocity field, while preserving the distance property. In fact such a velocity field exists and efficient algorithms are known to construct it, see e.g. [1]. Such velocity fields are known as *extension velocities*, as they are constructed extending the velocity prescribed on the interface to the whole domain.
Alternatively, we can still use the fluid velocity $\boldsymbol{u}$ for advecting the level set function ϕ, and intervene when $|\boldsymbol{\nabla}\phi|$ becomes too large or too small. The action to be taken in this case is known as *reinitialization*, as the procedure is partially the same as for initialization with the initial condition. Suppose we decide to reinitialize at time $t = t_r$:

1. Given $\phi(\cdot, t_r)$, find $\Gamma(t_r) = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t_r) = 0\}$.
2. Replace $\phi(\cdot, t_r)$ by $\mathrm{sdist}(\cdot, \Gamma(t_r))$.

Interestingly, it turns out that the problem of finding the extension velocity is closely related to the problem of reinitializing ϕ to a signed distance function. The same algorithms can be used and the same computational cost has to be expected. Two conceptual differences favour the reinitialization approach though: Firstly, the extension velocities have to be computed at every timestep, whereas reinitialization can be performed only when necessary, which

results in a global reduction of the computational costs. Secondly, the approximated extension velocities will only approximately conserve the distance property and may not guarantee that reinitialization is unnecessary.
Algorithmic details about the efficient construction of an approximation to the signed distance function, especially for the three-dimensional case, can be found in Section 3.5.

1.3 Well-Posedness

We now take the questions of well-posedness of the problem described by the model equations, and of existence and uniqueness of its solutions.
A global existence result for the coupled problem (1.1)-(1.2)-(1.3) with $\boldsymbol{f} = \rho \boldsymbol{g}$ and $\sigma = 0$ has been proven by Lions [70]. This proof requires Ω to be a smooth, bounded, connected open subset of $\mathbb{R}^d$, and that homogeneous Dirichlet boundary conditions (i.e., with $\boldsymbol{g}_D = \boldsymbol{0}$) are imposed on the whole boundary. If the initial and source data satisfy

$$\begin{aligned} \rho_0 &\geq 0 \quad \text{a.e. in } \Omega \\ \rho_0 &\in L^\infty(\Omega) \\ \rho_0 \boldsymbol{u}_0 &\in L^2(\Omega)^d \\ \rho_0 \left|\boldsymbol{u}_0\right|^2 &\in L^1(\Omega) \\ \boldsymbol{g} &\in L^2(\Omega \times (0,T))^d, \end{aligned}$$

then there exist global solutions which satisfy

$$\begin{aligned} \rho &\in L^\infty(\Omega \times (0,T)) \\ \boldsymbol{u} &\in L^2(0,T; H_0^1(\Omega))^d \\ \rho \left|\boldsymbol{u}\right|^2 &\in L^\infty(0,T; L^1(\Omega)) \\ \boldsymbol{\nabla u} &\in L^2(\Omega \times (0,T)) \\ \rho &\in C([0,\infty); L^p(\Omega)) \quad \forall p \in [1,\infty) \end{aligned}$$

These solutions are called *weak* solutions, as nothing more than the above weak statements is known. Specially nothing is known about the regularity of the pressure field p.
However, if $\rho_0 > 0$ then it is proven [70] that there is a short time smooth strong solution to which all weak solutions are equal.
Another result by Tanaka [103] treats the case where the surface tension coefficient σ is different from zero but constant. Under some (stronger) regularity assumptions on the initial data, it has been proven that a global solution exists for sufficiently small initial data and external forces. Moreover, local (in time) uniqueness is proved.
This situation is not very satisfactory, however it does not prevent us from developing well-posed numerical models approximating the mathematical models and providing solutions which are in good agreement with physical observations.

Chapter 2

Finite Element Methods with Interior Penalty Stabilization

2.1 Preliminaries

In this chapter, we introduce different finite element methods for incompressible flow in general, and for the Navier-Stokes equations with discontinuous density and viscosity in particular. In Section 2.2, we present an interior penalty stabilized finite element method for the Oseen equation. Thanks to equal order interpolation, this method is well suited for incompressible flow computations, specially for two fluid flows without surface tension. In Section 2.3, we recall the most important theoretical results known for continuous interior penalty methods for incompressible flows with constant density and viscosity. We also give a short overview of known results for finite element methods applied to flows with discontinuous density and viscosity. We then present in Section 2.4 the Taylor-Hood finite element scheme. It will serve both for comparison in general, and as an alternative in the case of two fluid flows with surface tension. A first step towards an analysis of an interior penalty finite element scheme for two fluid flow is presented in Section 2.5, where we consider the two dimensional stationary Stokes equation. Finally, an interior penalty stabilized finite element formulation for the advection-reaction equation is presented in Section 2.6. This scheme will be used for the transport equation of the interface in two fluid flow computations.

2.1.1 Motivation

The well posedness of approximations of the Stokes and Oseen equations using standard Galerkin methods depends on the inf-sup condition, also called Ladyshenskaja-Babuska-Brezzi (LBB) condition. Upon discretization with finite elements, this calls for different discrete finite element spaces for velocities and pressure (see [13]). It is often more convenient to use the same interpolation spaces for velocity and pressure, which means that the stability must be recovered in some other fashion. *Stabilized* finite element methods achieve this by adding terms to the standard Galerkin formulation. In the case of high Reynolds numbers, i.e., when the convection dominates over the viscous effects, other numerical instabilities occur even for inf-sup stable element spaces. These instabilities are related to the convection itself as well as to the incompressibility condition and can be controlled by stabilized methods as well.
The popular streamline upwind Petrov-Galerkin (SUPG) method has been introduced by

Hughes et al. [55]. Its interpretation in the context of variational multiscale methods and of residual-free bubbles has been given in [14]. Applied to both the velocities and the pressure, the SUPG method has been analyzed by Hansbo and Szepessy in [46], by Franca and Frey in [37] and by Tobiska and Verfürth in [109]. Despite its success from both theoretical and practical points of view, this method has some drawbacks, namely the introduction of artificial boundary conditions on velocities and pressure (see e.g. [33] or [6]), and the additional coupling of time-derivatives with space-derivatives, restricting the choice of time discretization and making its analysis difficult.
Several alternative stabilization techniques have been developed recently, amongst which the continuous interior penalty method (see [18], [19], [17], [16]). In the latter method, the idea is to add a least squares penalization term on the jump of the gradient between adjacent elements.
In this chapter, we introduce various finite element methods. In Section 2.2, we formulate continuous interior penalty methods applicable to two-fluid flows. Section 2.3 recalls known theoretical results for this and relevant related methods. For cases where inf-sup stable elements are preferable and for comparison of numerical results, we briefly introduce the Taylor-Hood formulation in Section 2.4. In Section 2.5, we present an original analysis for an interior penalty method applied to the stationary Stokes problem with variable viscosity. Finally, we introduce in Section 2.6 a continuous interior penalty method for the advection reaction equation, which will be applied for the discrete interface evolution model.

2.1.2 Time Discretization and Linearization of Flow Equations

In order to apply the finite element method to the flow equations from Chapter 1, we carry out different steps to reformulate the problem in a suitable and reasonably general way.
In a first step, we discretize the flow equations in time and then linearize them. For time discretization, we choose the backward differencing scheme of order 2 (BDF2) with a fixed timestep Δt. We express the time derivative of a function $y(t)$ as

$$\frac{\mathrm{d}y}{\mathrm{d}t}(t^{n+1}) = \frac{1}{2\Delta t}(3y(t^{n+1}) - 4y(t^n) + y(t^{n-1})) + \mathcal{O}(\Delta t^2),$$

where we use the notation t^k for $k\,\Delta t$ and the Landau symbol $\mathcal{O}(\xi)$ for an expression whose absolute value is bounded from above by a constant times ξ. The BDF2 scheme for the ordinary differential equation $\mathrm{d}y/\mathrm{d}t = b$ then reads

$$\frac{1}{2\Delta t}(3y(t^{n+1}) - 4y(t^n) + y(t^{n-1})) = b(t^{n+1}). \tag{2.1}$$

For the motivation of this choice, see Subsection 3.1.1. Note that most other common time discretization schemes will lead to equations of the same form as below, only the coefficients will be different. Applying (2.1) to the flow equations (1.1) and (1.2), we obtain

$$\alpha^{n+1}\boldsymbol{u}^{n+1} + (\boldsymbol{\beta}^{n+1}\cdot\boldsymbol{\nabla})\boldsymbol{u}^{n+1} - \boldsymbol{\nabla}\cdot(2\mu(t^{n+1})\mathbf{D}(\boldsymbol{u}^{n+1})) + \boldsymbol{\nabla}p^{n+1} = \boldsymbol{f}^{n+1} \quad \text{in } \Omega, \tag{2.2}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u}^{n+1} = 0 \quad \text{in } \Omega, \tag{2.3}$$

with

$$\begin{aligned}
\boldsymbol{u}^n &= \boldsymbol{u}(t^n),\\
p^{n+1} &= p(t^{n+1}),\\
\alpha^{n+1} &= \frac{3\rho(t^{n+1})}{2\Delta t} > 0,\\
\boldsymbol{\beta}^{n+1} &= \rho(t^{n+1})\hat{\boldsymbol{u}}^{n+1} \qquad \text{and}\\
\boldsymbol{f}^{n+1} &= \boldsymbol{f}(t^{n+1}) + \frac{\rho(t^{n+1})}{2\Delta t}(4\boldsymbol{u}^n - \boldsymbol{u}^{n-1}).
\end{aligned}$$

In the expression $\boldsymbol{\beta}^{n+1}$ we have replaced $\boldsymbol{u}^{n+1}$ by a suitable approximation $\hat{\boldsymbol{u}}^{n+1}$ in order to linearize the equation. To obtain second order accuracy, an extrapolated value $\hat{\boldsymbol{u}}^{n+1} = 2\boldsymbol{u}^n - \boldsymbol{u}^{n-1}$ can be used for example. See Subsection 3.1.1 for a detailed discussion on strategies for the choice of $\hat{\boldsymbol{u}}^{n+1}$.
For the rest of the chapter, we shall drop the superscripts indicating time levels for better readability. The semi-discretized and linearized flow equations take then the form of an *Oseen* problem:

$$\alpha\boldsymbol{u} + (\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \boldsymbol{\nabla}\cdot(2\mu\mathbf{D}(\boldsymbol{u})) + \boldsymbol{\nabla}\, p = \boldsymbol{f} \quad \text{in } \Omega, \tag{2.4}$$

$$\boldsymbol{\nabla}\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega, \tag{2.5}$$

where $\alpha > 0$ is a given positive scalar field and $\boldsymbol{\beta}$ and $\boldsymbol{f}$ are given vector fields. These equations have to be completed by boundary conditions as presented in Subsection 1.1.2.
With similar operations, many other flow model problems can be cast in the form of the Oseen equations, allowing for a general treatment regarding both analysis and development of numerical schemes.

2.1.3 Weak Formulation

In order to derive the weak formulation, we first need to introduce some notation for function spaces. The space of p-integrable functions on $\Omega \subset \mathbb{R}^d$, denoted by $L^p(\Omega)$, $1 \le p < \infty$ is the space of functions v for which $\int_\Omega |v(\boldsymbol{x})|^p \,\mathrm{d}\boldsymbol{x} < \infty$. The space $L^\infty(\Omega)$ is the space of functions v for which $\operatorname{ess\,sup}_{\boldsymbol{x}\in\Omega} |v(\boldsymbol{x})| < \infty$. The Sobolev space $W^{m,p}(\Omega)$ with $m \ge 0$ and $p \ge 1$ is defined as the space of functions v in $L^p(\Omega)$ with all partial derivatives (in distributional sense) up to order m in $L^p(\Omega)$. We use the standard notation $H^m(\Omega) = W^{m,2}(\Omega)$. The norm of $H^m(\Omega)$ is denoted by $\|\cdot\|_{m,\Omega}$ and defined by

$$\|v\|_{m,\Omega}^2 := \sum_{k_1=0}^{m} \cdots \sum_{k_d=0}^{m} \int_\Omega \left| \partial_{x_1}^{k_1} \ldots \partial_{x_d}^{k_d}\, v \right|^2 \mathrm{d}\boldsymbol{x}.$$

The seminorm of $H^m(\Omega)$ is denoted by $|\cdot|_{m,\Omega}$ and defined by $|v|_{m,\Omega}^2 = \|v\|_{m,\Omega}^2 - \|v\|_{m-1,\Omega}^2$, $m \ge 1$. Moreover, we define the spaces

$$\begin{aligned}
H_0^1(\Omega) &= \{w \in H^1(\Omega) : w|_{\partial\Omega} = 0\} \quad \text{and}\\
L_0^2(\Omega) &= \{w \in L^2(\Omega) : \int_\Omega w \,\mathrm{d}x = 0\}.
\end{aligned}$$

For L^2 scalar products on Ω of scalars p, q, vectors $\boldsymbol{u}, \boldsymbol{v}$ and rank two tensors $\mathbf{A}, \mathbf{B}$, we use the notation

$$(p,q) = \int_\Omega p\, q \,\mathrm{d}x, \qquad (\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}x \qquad \text{and} \qquad (\mathbf{A},\mathbf{B}) = \int_\Omega \mathbf{A}:\mathbf{B}\,\mathrm{d}x,$$

where $\mathbf{A}:\mathbf{B}$ denotes the Frobenius inner product

$$\mathbf{A}:\mathbf{B} = \sum_{i,j=1}^{d} (\mathbf{A})_{ij}\,(\mathbf{B})_{ij}.$$

For scalar products on boundaries and interfaces (i.e., lines for $d=2$ and surfaces for $d=3$), we write

$$\langle p,q\rangle_\gamma = \int_\gamma p\, q \,\mathrm{d}s \qquad \text{and} \qquad \langle \boldsymbol{u},\boldsymbol{v}\rangle_\gamma = \int_\gamma \boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}s.$$

We now derive the weak formulation for the Oseen problem (2.4)-(2.5). We partition $\partial\Omega$ into three subsets with different boundary conditions: $\partial\Omega = \Gamma_D \cup \Gamma_N \cup \Gamma_R$, requiring $\partial\Omega \setminus \Gamma_D \neq \emptyset$. We have

- on Γ_D: homogeneous Dirichlet boundary conditions, i.e., (1.4) with $\boldsymbol{g}_D|_{\Gamma_D} = 0$;
- on Γ_N: Neumann boundary conditions, i.e., (1.5);
- on Γ_R: mixed Robin boundary conditions with homogeneous normal part, i.e., (1.6)-(1.7) with $(\boldsymbol{g}_D\cdot\boldsymbol{n})|_{\Gamma_R} = 0$ and $\omega \in [0,1)$. With $\omega = 0$ we also cover the free slip/symmetry case, whereas the case of $\omega = 1$ is covered by the Dirichlet boundary conditions on Γ_D.

To obtain the weak formulation, we first multiply formally (2.4) by a function $\boldsymbol{v} \in V := \{\boldsymbol{w} \in [H^1(\Omega)]^d : \boldsymbol{w}|_{\Gamma_D} = \mathbf{0} \text{ and } (\boldsymbol{w}\cdot\boldsymbol{n})|_{\Gamma_R} = 0\}$ and integrate by parts over $\Omega^* = \Omega^- \cup \Omega^+$. We cannot integrate by parts over Ω, because the integrands do not have the necessary regularity. We obtain

$$\sum_{i\in\{+,-\}} \Big(\int_{\Omega^i} \big(\alpha\boldsymbol{u}\cdot\boldsymbol{v} + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u})\cdot\boldsymbol{v} + 2\mu\mathbf{D}(\boldsymbol{u}):\mathbf{D}(\boldsymbol{v}) - p\,\boldsymbol{\nabla}\cdot\boldsymbol{v}\big)\mathrm{d}\boldsymbol{x} \\ - \int_{\partial\Omega^i} \big((2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n})\cdot\boldsymbol{v} - p\boldsymbol{n}\cdot\boldsymbol{v}\big)\mathrm{d}s \Big) = \sum_{i\in\{+,-\}} \int_{\Omega^i} \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}x.$$

We note that the sum of two integrals over Ω^+ and Ω^- can be written as integral over Ω^*, or as integral over Ω, because $\Omega \setminus \Omega^* = \Gamma$ has zero measure. The subdomain boundaries form together the domain boundary and the interface: $\partial\Omega^+ \cup \partial\Omega^- = \partial\Omega \cup \Gamma$. So the sum of the integrals over the subdomain boundaries can be rewritten as an integral over the boundary $\partial\Omega$ plus two integrals over the interface Γ. The integrands of the latter may seem equal at first, but $\mathbf{D}(\boldsymbol{u})$ and p have to be interpreted as traces on $\partial\Omega^i$, which may be have different limits from either side, and the outer unit normal $\boldsymbol{n}$ has opposite direction on $\partial\Omega^+$ with respect

to $\partial\Omega^-$. We therefore obtain a jump term on the interface, subtracting the limits from both sides:

$$(\alpha\boldsymbol{u},\boldsymbol{v}) + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{v}) + (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v})) - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v}) \\ - \langle 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n},\boldsymbol{v}\rangle_{\partial\Omega} + \langle[\![2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}_\Gamma - p\,\boldsymbol{n}_\Gamma]\!]_\Gamma,\boldsymbol{v}\rangle_\Gamma = (\boldsymbol{f},\boldsymbol{v}).$$

In a second step, we multiply formally (2.5) by a function $q \in L^2(\Omega) = Q$ and integrate over Ω:

$$(\boldsymbol{\nabla}\cdot\boldsymbol{u},q) = 0.$$

Adding the two results leads to the problem of finding $(\boldsymbol{u},p) \in V \times Q$ such that

$$(\alpha\boldsymbol{u},\boldsymbol{v}) + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{v}) + (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v})) - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v}) + (\boldsymbol{\nabla}\cdot\boldsymbol{u},q) \\ - \langle 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n},\boldsymbol{v}\rangle_{\partial\Omega} + \langle[\![2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}_\Gamma - p\,\boldsymbol{n}_\Gamma]\!]_\Gamma,\boldsymbol{v}\rangle_\Gamma = (\boldsymbol{f},\boldsymbol{v}). \quad (2.6)$$

Using (1.5), (1.10) and the fact that $\boldsymbol{v}|_{\Gamma_D} = \mathbf{0}$ because $\boldsymbol{v} \in V$, we can write

$$(\alpha\boldsymbol{u},\boldsymbol{v}) + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{v}) + (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v})) - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v}) + (\boldsymbol{\nabla}\cdot\boldsymbol{u},q) \\ - \langle 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n},\boldsymbol{v}\rangle_{\Gamma_R} = (\boldsymbol{f},\boldsymbol{v}) + \langle\boldsymbol{g}_N,\boldsymbol{v}\rangle_{\Gamma_N} + \langle\sigma\kappa\boldsymbol{n}_\Gamma,\boldsymbol{v}\rangle_\Gamma. \quad (2.7)$$

Note that due to the properties of V, we have

$$\int_\Gamma [\![\boldsymbol{u}]\!]_\Gamma\cdot\boldsymbol{v}\,\mathrm{d}s = 0 \quad \forall\boldsymbol{v}\in V,$$

as $\boldsymbol{u} \in V$, i.e., condition (1.9) is already satisfied weakly. It now remains to treat the boundary Γ_R. For that, we decompose $2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n}$ and $\boldsymbol{v}$ into a normal and a tangential component, and use the fact that $(\boldsymbol{u}\cdot\boldsymbol{n})|_{\Gamma_R} = (\boldsymbol{v}\cdot\boldsymbol{n})|_{\Gamma_R} = 0$ as $\boldsymbol{u},\boldsymbol{v} \in V$. Moreover we apply equation (1.7) divided by $1-\omega$ to obtain

$$-\langle 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n} - p\,\boldsymbol{n},\boldsymbol{v}\rangle_{\Gamma_R} = \langle\omega^*\boldsymbol{u},\boldsymbol{v}\rangle_{\Gamma_R} - \langle\omega^*\boldsymbol{g}_D,\boldsymbol{v}\rangle_{\Gamma_R}, \quad (2.8)$$

with $\omega^* = \omega C_\tau/(1-\omega)$, which is a positive function. Substituting (2.8) into (2.7), we obtain the variational problem:

$$\text{Find } (\boldsymbol{u},p)\in V\times Q \text{ such that} \qquad B[(\boldsymbol{u},p),(\boldsymbol{v},q)] = f(\boldsymbol{v}) \qquad \forall(\boldsymbol{v},q)\in V\times Q,$$

where

$$B[(\boldsymbol{u},p),(\boldsymbol{v},q)] = (\alpha\boldsymbol{u},\boldsymbol{v}) + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{v}) + (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v})) + \langle\omega^*\boldsymbol{u},\boldsymbol{v}\rangle_{\Gamma_R} \\ - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v}) + (q,\boldsymbol{\nabla}\cdot\boldsymbol{u}),$$
$$f(\boldsymbol{v}) = (\boldsymbol{f},\boldsymbol{v}) + \langle\boldsymbol{g}_N,\boldsymbol{v}\rangle_{\Gamma_N} + \langle\sigma\kappa\boldsymbol{n}_\Gamma,\boldsymbol{v}\rangle_\Gamma + \langle\omega^*\boldsymbol{g}_D,\boldsymbol{v}\rangle_{\Gamma_R}.$$

2.1.4 Triangulation and Discrete Spaces

Let $\{\mathcal{T}_h\}_h$ be a family of triangulations of the domain Ω. By a triangulation $\mathcal{T}_h$, we understand a partition of Ω by closed simplices K such that

$$\bar\Omega = \bigcup_{K\in\mathcal{T}_h} K,$$

where $\bar{\Omega}$ denotes the closure of Ω. We suppose that for all $K, K^* \in \mathcal{T}_h$, $\bar{K} \cap \bar{K}^*$ is either a corner, an entire edge or an entire face of both K and K^*. For each triangulation $\mathcal{T}_h$, the subscript h refers to the level of refinement defined by

$$h := \max_{K \in \mathcal{T}_h} h_K,$$

where

$$h_K := \operatorname{diam}(K) := \sup_{\boldsymbol{x},\boldsymbol{y} \in K} |\boldsymbol{x} - \boldsymbol{y}|.$$

Moreover we will assume that the family $\mathcal{T}_h$ is quasiuniform, i.e.,

$$\frac{h_K}{\rho_k} < C_R \quad \text{and} \quad h_K \geq C_U\, h \qquad \forall K \in \mathcal{T}_h, \quad \forall \mathcal{T}_h \in \{\mathcal{T}_h\},$$

where ρ_K is the diameter of the largest ball inscribed in K, and $C_R, C_U > 0$ are fixed constants. In what follows, the word *faces* refers to edges (lines) for $d = 2$ and to faces (triangles) for $d = 3$. We denote by Γ_I the union of all interior faces, $\Gamma_I = \bigcup_{K \in \mathcal{T}} \partial K \setminus \partial\Omega$, and by $[\![\cdot]\!]_f$ the jump of a quantity across an internal face $f \subset \Gamma_I$, defined as follows:

$$[\![v]\!]_f(\boldsymbol{x}) = \lim_{\varepsilon \to 0^+} v(\boldsymbol{x} + \varepsilon \boldsymbol{n}_f) - v(\boldsymbol{x} - \varepsilon \boldsymbol{n}_f), \tag{2.9}$$

where $\boldsymbol{n}_f$ is a fixed but arbitrary normal unit vector on the face f and $\boldsymbol{x} \in f$. The diameter of a face f is denoted by h_f.
On these triangulations we introduce the spaces of piecewise constants and of continuous piecewise polynomial functions of degree k

$$\begin{aligned} V_h^0 &= \{v \in L^2(\Omega) : v|_K \in \mathbb{P}_0(K) \quad \forall K \in \mathcal{T}_h\}, \\ V_h^k &= \{v \in H^1(\Omega) : v|_K \in \mathbb{P}_k(K) \quad \forall K \in \mathcal{T}_h\} \qquad k = 1, 2, 3, \ldots, \\ \boldsymbol{V}_h^k &= [V_h^k]^d. \end{aligned}$$

2.2 Interior Penalty Formulation of Flow Equations

In this section, we consider a finite element method with interior penalty stabilization for the incompressible Navier-Stokes equations. This method was introduced by Burman and Hansbo in [18] for pure transport problems or convection-dominated problems. Pressure stabilization for the Stokes problem was then considered in [19] and the Oseen problem was analyzed in [17]. Our scheme is based on the latter, applicable to the time dependent Navier-Stokes equations after time semidiscretization and linearization. Covering the special case of the inviscid limit $\mu \to 0$, there is an analysis [16] of an interior penalty stabilization method for the fully nonlinear and time dependent Navier-Stokes equations. It requires stronger stabilization of the velocities that can be avoided in the viscous case. See Subsection 2.3.1 for a short presentation of these results.
For the finite element formulation, we will consider also inhomogeneous Dirichlet boundary conditions. Moreover, we choose to impose all boundary conditions weakly, following the approach from [17] and [58]. We partition $\partial\Omega$ into two subsets $\partial\Omega = \Gamma_N \cup \Gamma_R$, incorporating the Dirichlet case in the mixed Robin case with $\omega = 1$. We formally proceed as in Subsection 2.1.3 with $\Gamma_D = \emptyset$ to obtain (2.7). The boundary Γ_R is however treated differently. Using

the mixed Robin boundary conditions (1.6)-(1.7), we obtain the following identities, which we will add to (2.7):

$$\langle C_n \boldsymbol{u}\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\Gamma_R} = \langle C_n \boldsymbol{g}_D\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\Gamma_R}, \tag{2.10}$$

$$\langle \omega C_\tau \mathbf{P}\boldsymbol{u}, \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R} + \langle (1-\omega)\mathbf{P}(2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}), \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R} = \langle \omega C_\tau \mathbf{P}\boldsymbol{g}_D, \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R}, \tag{2.11}$$

$$-\langle \omega \mathbf{P}\boldsymbol{u}, \mathbf{P}(2\mu\mathbf{D}(\boldsymbol{v})\boldsymbol{n})\rangle_{\Gamma_R} = -\langle \omega \mathbf{P}\boldsymbol{g}_D, \mathbf{P}(2\mu\mathbf{D}(\boldsymbol{v})\boldsymbol{n})\rangle_{\Gamma_R}, \tag{2.12}$$

where $\mathbf{P} = \mathbf{I} - \boldsymbol{n}\otimes\boldsymbol{n}$ denotes the projector on the tangent line in two dimensions, and on the tangent plane in three dimensions.
Equation (2.10) enforces (1.6), whereas (2.11) covers (1.7). The third equation (2.12) is added for preserving symmetry in the diffusive terms. We choose

$$C_n = \gamma_n \max\{|\boldsymbol{\beta}|, \mu/h\}, \text{ and}$$
$$C_\tau = \gamma_\tau \frac{\mu}{h} + \max\{-\boldsymbol{\beta}\cdot\boldsymbol{n}, 0\},$$

following [17] and [58]. The penalty parameters γ_n and γ_τ are dimensionless constants, and the physically correct scaling is recovered by using μ and β. The terms involving β allow to capture the case of dominant convection.
We add (2.10)-(2.12) to (2.7), and define the space W_h^k as the product $\boldsymbol{V}_h^k \times V_h^k$. Our finite element scheme for the Oseen equation then reads
Find $(\boldsymbol{u}_h, p_h) \in W_h^k$ such that

$$B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{v}_h, q_h)] = f_h(\boldsymbol{v}_h) \qquad \forall (\boldsymbol{v}_h, q_h) \in W_h^k, \tag{2.13}$$

where

$$\begin{aligned} B_h[(\boldsymbol{u},p),(\boldsymbol{v},q)] &= a_h(\boldsymbol{u},\boldsymbol{v}) + b_h(p,\boldsymbol{v}) - b_h(q,\boldsymbol{u}) \\ &\quad + j_{\boldsymbol{\beta}}(\boldsymbol{u},\boldsymbol{v}) + j_{\mathrm{div}}(\boldsymbol{u},\boldsymbol{v}) + j_p(p,q), \end{aligned} \tag{2.14}$$

$$\begin{aligned} a_h(\boldsymbol{u},\boldsymbol{v}) &= (\alpha\boldsymbol{u},\boldsymbol{v}) + ((\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u},\boldsymbol{v}) + (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v})) \\ &\quad - \langle \omega\ \mathbf{P}(2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}), \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R} - \langle \omega\ \mathbf{P}\boldsymbol{u}, \mathbf{P}(2\mu\mathbf{D}(\boldsymbol{v})\boldsymbol{n})\rangle_{\Gamma_R} \\ &\quad + \langle C_n\boldsymbol{u}\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\Gamma_R} + \langle \omega C_\tau \mathbf{P}\boldsymbol{u}, \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R}, \end{aligned} \tag{2.15}$$

$$b_h(p,\boldsymbol{v}) = -(p, \boldsymbol{\nabla}\cdot\boldsymbol{v}), \tag{2.16}$$

$$j_{\boldsymbol{\beta}}(\boldsymbol{u},\boldsymbol{v}) = \left\langle \gamma_{\boldsymbol{\beta}}\ \frac{h_f^2}{|\boldsymbol{\beta}|}[\![(\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u}]\!]_f, [\![(\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{v}]\!]_f \right\rangle_{\Gamma_I}, \tag{2.17}$$

$$j_{\mathrm{div}}(\boldsymbol{u},\boldsymbol{v}) = \langle \gamma_{\mathrm{div}}\ h_f^2\,|\boldsymbol{\beta}|\ [\![\boldsymbol{\nabla}\cdot\boldsymbol{u}]\!]_f, [\![\boldsymbol{\nabla}\cdot\boldsymbol{v}]\!]_f\rangle_{\Gamma_I}, \tag{2.18}$$

$$j_p(p,q) = \left\langle \gamma_p \frac{h_f^3}{\max\{h_f\,|\boldsymbol{\beta}|, \mu\}}\ [\![\boldsymbol{\nabla}p]\!]_f, [\![\boldsymbol{\nabla}q]\!]_f \right\rangle_{\Gamma_I} \tag{2.19}$$

and

$$\begin{aligned} f_h(\boldsymbol{v}) &= (\boldsymbol{f},\boldsymbol{v}) + \langle \boldsymbol{g}_N,\boldsymbol{v}\rangle_{\Gamma_N} + \langle \sigma\kappa\boldsymbol{n}_\Gamma,\boldsymbol{v}\rangle_\Gamma - \langle \omega\mathbf{P}\boldsymbol{g}_D, \mathbf{P}(2\mu\mathbf{D}(\boldsymbol{v})\boldsymbol{n})\rangle_{\Gamma_R} \\ &\quad + \langle C_n\boldsymbol{g}_D\cdot\boldsymbol{n}, \boldsymbol{v}\cdot\boldsymbol{n}\rangle_{\Gamma_R} + \langle \omega C_\tau\mathbf{P}\boldsymbol{g}_D, \mathbf{P}\boldsymbol{v}\rangle_{\Gamma_R}, \end{aligned} \tag{2.20}$$

where $\gamma_{\boldsymbol{\beta}}$, γ_{div}, and γ_p are dimensionless positive constants.
We have added three gradient jump terms. They serve three purposes:

1. The term $j_{\boldsymbol{\beta}}$ (2.17) stabilizes the convective terms, necessary at high Reynolds numbers.

2. The term j_{div} (2.18) gives additional control of the incompressibility condition, also necessary at high Reynolds numbers.

3. The term j_p (2.19) makes the discretization inf-sup stable for equal order interpolation spaces.

The parameters $\gamma_{\boldsymbol{\beta}}$ and γ_{div} can therefore be set to zero for low Reynolds number computations. The well posedness of this scheme is proven in [17] for constant density and viscosity. We recall this and other results in Subsection 2.3.1.

2.3 Known Theoretical Results for Flow Equations

In this section, we recall the most important known theoretical results for continuous interior penalty finite elements and for finite elements for discontinuous density and viscosity. A first step towards the combination of the two can be found in Section 2.5.

2.3.1 Known Results for Constant Density and Viscosity

In this section, we shortly revisit the known results of continous interior penalty finite elements for different flow models with constant density and viscosity.

Stokes Equations

We first consider the generalized Stokes equations with homogeneous Dirichlet boundary conditions

$$\begin{aligned} \alpha \boldsymbol{u} - \mu \Delta \boldsymbol{u} + \boldsymbol{\nabla} p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \boldsymbol{\nabla} \cdot \boldsymbol{u} &= g \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \partial\Omega, \end{aligned}$$

with α and μ real positive constants.
For these equations, Burman and Hansbo propose in [19] a continuous finite element scheme with interior penalty very similar to (2.13) with $\boldsymbol{\beta} = 0$. The main differences are the slightly different weights in the penalty terms and the strong imposition of the boundary conditions. They prove the optimal a priori estimate

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{0,\Omega} + h(\|\boldsymbol{\nabla}(\boldsymbol{u} - \boldsymbol{u}_h)\|_{0,\Omega} + \|p - p_h\|_{0,\Omega}) \leq Ch^2(\|\boldsymbol{u}\|_{2,\Omega} + \|p\|_{1,\Omega})$$

for the discrete solution $(\boldsymbol{u}_h, p_h) \in W_h^1$. They also show numerical evidence that no artificial boundary layers appear in the approximate pressure unlike in some other stabilized methods such as Galerkin least squares (GLS).

Oseen Equations

In [17], the authors generalize the continuous interior penalty method to the Oseen problem (2.4)-(2.5) with weakly imposed homogeneous Dirichlet boundary conditions on the whole

boundary. The coefficients α and μ are real positive constants whereas the convective field $\boldsymbol{\beta}$ is assumed to be in $[W^{1,\infty}(\Omega)]^d$.
Beside the boundary conditions, the proposed finite element scheme is identical to the one presented in Section 2.2, except for additional terms and slightly different scaling of jump penalty terms in order to achieve a priori estimates independent of Reynolds number.
The discrete problem is well posed (see [17]), i.e., there exists a unique discrete solution. Supposing that the mesh is locally quasi-uniform and that it resolves well the variations of $\boldsymbol{\beta}$, and that the exact solution $(\boldsymbol{u}, p)$ belongs to $[H^2(\Omega)]^d \times H^1(\Omega)$, the following a priori estimates for the discrete solution $(\boldsymbol{u}_h, p_h) \in W_h^k$ are proven:

$$\begin{aligned}\|\boldsymbol{u} - \boldsymbol{u}_h\|_{1,\Omega} &\leq Ch \|\boldsymbol{u}\|_{2,\Omega} + Ch^{1/2} \|p\|_{1,\Omega}\,,\\ \|p - p_h\|_{0,\Omega} &\leq Ch \|\boldsymbol{u}\|_{2,\Omega} + Ch^{1/2} \|p\|_{1,\Omega}\,.\end{aligned}$$

For the low Reynolds number case, it is also proven that

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{0,\Omega} \leq Ch^2(\|\boldsymbol{u}\|_{2,\Omega} + \|p\|_{1,\Omega}), \tag{2.21}$$

which is optimal for $k = 1$, and

$$\|p - p_h\|_{0,\Omega} \leq Ch(\|\boldsymbol{u}\|_{2,\Omega} + \|p\|_{1,\Omega}). \tag{2.22}$$

Navier-Stokes Equations

In [16], Burman and Fernández consider the fully nonlinear and time-dependent Navier-Stokes equations:

$$\begin{aligned}\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \boldsymbol{\nabla}\cdot(2\mu\mathbf{D}(\boldsymbol{u})) + \boldsymbol{\nabla} p &= \boldsymbol{f} \quad \text{in } \Omega\times(0,T),\\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega\times(0,T),\\ \boldsymbol{u} &= \mathbf{0} \quad \text{on } \partial\Omega\times(0,T),\\ \boldsymbol{u}|_{t=0} &= \boldsymbol{u}_0 \quad \text{in } \Omega\end{aligned}$$

where μ is a real positive constant. These equations correspond to (1.1)-(1.2) with $\rho = 1$.
The scheme proposed in [16] differs from ours, beside being still continuous in time and fully non-linear, in adding to the bilinear form the terms

$$\frac{1}{2}((\boldsymbol{\nabla}\cdot\boldsymbol{\beta})\boldsymbol{u}, \boldsymbol{v}) - \frac{1}{2}\langle(\boldsymbol{\beta}\cdot\boldsymbol{n})\boldsymbol{u}, \boldsymbol{v}\rangle_{\partial\Omega}$$

to counter effects of insufficient control of the divergence free condition and to ensure coercivity while remaining strongly consistent. Moreover, the interior penalty term for the velocities takes the form

$$\langle(\gamma + |\boldsymbol{\beta}\cdot\boldsymbol{n}|^2)h^2[\![\boldsymbol{\nabla}\boldsymbol{u}]\!]_f, [\![\boldsymbol{\nabla}\boldsymbol{v}]\!]_f\rangle_{\Gamma_I}.$$

Under the regularity assumptions

$$\begin{aligned}&\boldsymbol{u} \in [L^\infty(0,T;W^{1,\infty}(\Omega)) \cap H^1(0,T;L^2(\Omega)) \cap L^\infty(0,T;H^r(\Omega))]^d \text{ and}\\ &p \in L^2(0,T;H^s(\Omega))\end{aligned}$$

with $r, s \geq 2$, the authors demonstrate that the discrete problem has a unique solution $(\boldsymbol{u}_h, p_h) \in C^1(0,T;\boldsymbol{V}_h^k) \times C^0(0,T;V_h^k)$. Under the same assumptions, they prove the quasi-optimal error estimate for the velocity approximation

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{L^\infty(0,T;L^2(\Omega))} \leq Ch^{k+1/2} \|(\boldsymbol{u}, p)\|_{L^2(0,T;H^{k+1}(\Omega))},$$

where C is a constant depending on $\boldsymbol{u}$ but not on μ. This result is of special interest for dominant convection, i.e., $|\boldsymbol{u}|\, h \gg \mu$. Interior penalty methods for the Navier-Stokes equations approximated with *discontinuous* pressure have been studied e.g. in [48] and [49].

2.3.2 Known Results for Variable Density and Viscosity

In our two-fluid problem, where density and viscosity are variable and discontinuous, the mesh is not fitted to the discontinuities of the coefficients in the exact solution. It is well known that for elliptic problems this leads to loss of accuracy and pollution effects perturbing the solution close to the front. This can be seen e.g. in an analysis by Ohmori and Saito [79] of a ($\mathbb{P}_1$ iso $\mathbb{P}_2$)-$\mathbb{P}_1$ scheme for the stationary Stokes problem in two dimensions, where they prove

$$\begin{aligned}
\|\boldsymbol{u} - \boldsymbol{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} \leq & Ch^\varepsilon(\|\boldsymbol{u}\|_{1+\varepsilon,\Omega} + \|p\|_{\varepsilon,\Omega}) \quad \varepsilon \in (0, 1/2) \quad \text{and} \\
\|\boldsymbol{u} - \boldsymbol{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega} \leq & Ch^\varepsilon(\|\boldsymbol{u}\|_{1+\varepsilon,\Omega^*} + \|p\|_{\varepsilon,\Omega^*}) \quad \varepsilon \in [1/2, 1].
\end{aligned}$$

To our knowledge, no theoretical results are known for continuous interior penalty schemes in this case. In section 2.5, we suggest and analyze an unfitted interior penalty scheme for the two dimensional stationary Stokes equation, which can be seen as a first step in this direction. Other methods have been developed and applied to flow problems with discontinuous coefficients:

- *Regularization of Density and Viscosity*
  The density and the viscosity can be regularized in order to improve the convergence rate. A detailed analysis of the quadrature errors for this approach is provided in [110].
- *Mesh Adaptation*
  Different strategies of mesh adaptation have been proposed ([26], [72], [73], [106], [107], [108]) to refine the triangulation near the interface and thereby reduce the amount of oscillations.
- *Extended Finite Elements (X-FEM)*
  An enrichment of the finite element space by the so called *extended finite element method* is suggested in [27] and in [42]. Similar to our scheme in Section 2.5, more degrees of freedom are introduced to recover convergence.

Finally, let us cite a different approach which consists in solving not the level set equation (1.15) but the mass conservation equation (1.3). The density is approximated in a finite element space, and the viscosity is expressed as a (linear) function of the density. Results can be found in [71] for $\rho_h \in V_h^0$ and in [102] for $\rho_h \in V_h^1$.

2.4 Taylor-Hood Formulation of Flow Equations

2.4.1 Motivation

We consider the well known Taylor-Hood finite elements (see, e.g., [91], [13]) for two reasons:

- When the surface tension is not zero, the exact solution of the pressure jumps at the interface (see [66], [68]). In this case the interior penalty scheme from Section 2.2 is not consistent because of the pressure gradient jump term. Because many other stabilized methods also require increased regularity of the pressure for their consistency, inf-sup stable methods like the Taylor-Hood scheme seem preferable in this case.
- It allows us to compare efficiency and accuracy of the interior penalty scheme with a well established method.

2.4.2 Weak Formulation

Taylor and Hood proposed in [104] a finite element scheme for the Navier-Stokes equations. They look for numerical solutions $(\boldsymbol{u}_h, p_h)$ in

$$X_{h,0}^2 = [\boldsymbol{V}_h^2 \cap [H_0^1(\Omega)]^d] \times [V_h^1 \cap L_0^2(\Omega)],$$

i.e., piecewise quadratic velocities and piecewise affine pressure.
This approach has been generalized to the space

$$X_{h,0}^k = [\boldsymbol{V}_h^k \cap [H_0^1(\Omega)]^d] \times [V_h^{k-1} \cap L_0^2(\Omega)], \quad k \geq 2,$$

which, applied to the Stokes problem, yields the optimal error estimate

$$\|\boldsymbol{u} - \boldsymbol{u}_h\|_{0,\Omega} + h(\|\boldsymbol{u} - \boldsymbol{u}_h\|_{1,\Omega} + \|p - p_h\|_{0,\Omega}) \leq c\, h^{k+1}(\|\boldsymbol{u}\|_{k+1,\Omega} + \|p\|_{k,\Omega}), \tag{2.23}$$

provided the exact solution $(\boldsymbol{u}, p)$ is smooth enough. Proofs and further insight can be found in [12] and [13].
In our comparative computations, we will apply exactly the same bilinear forms as for the interior penalty method, allowing for better comparability. Because we impose boundary conditions weakly, we use the spaces

$$X_h^k = \boldsymbol{V}_h^k \times V_h^{k-1}, \quad k \geq 2.$$

Our finite element scheme on this space then reads as follows:
Find $(\boldsymbol{u}_h, p_h) \in X_h^k$ such that

$$B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{v}_h, q_h)] = f_h(\boldsymbol{v}_h) \quad \forall (\boldsymbol{v}_h, q_h) \in X_h^k,$$

where B_h is defined by (2.14) and f_h is defined by (2.20). Note that due to the stability of the finite element space, we can (and will) choose $\gamma_p = 0$ in j_p (2.19). All interior penalty terms can therefore be neglected for the Taylor-Hood space and one obtains, except for the weak imposition of the boundary conditions, a standard finite element scheme like the original one from [104].

2.5 An Unfitted Scheme for the Stokes Problem

Since the position of the front is known only through intermediate of the level set function ϕ, the mesh is not fitted to the discontinuities of the coefficients in the exact solution. It is well known that for elliptic problems this leads to loss of convergence and pollution effects perturbing the solution close to the front. In this section, taken from the forthcoming paper [20],

our aim is to address the latter question only. Recently, the elliptic problem was successfully solved introducing a discontinuous approximation over the interface, thus recovering optimal order convergence (see [45]). We extend this technique to the two dimensional stationary Stokes equation with discontinuous density and viscosity, serving here as a model problem for the two-fluid equations from Chapter 1. We show that our formulation, which is well suited for the non-stationary case, satisfies the inf-sup condition and we prove an a priori estimate.

2.5.1 The Stationary Problem

Let Ω be a bounded domain in $\mathbb{R}^2$, with convex polygonal boundary $\partial\Omega$ and an internal smooth boundary Γ dividing Ω into two open sets called here Ω_1 and Ω_2 (instead of Ω^+ and Ω^-). The Stokes equation that we propose as a model problem is given by

$$\begin{aligned} -\boldsymbol{\nabla}\cdot(2\mu_i\mathbf{D}(\boldsymbol{u}))+\boldsymbol{\nabla}p &= \boldsymbol{f} \quad \text{in } \Omega_i,\ i=1,2 \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \boldsymbol{u} &= \mathbf{0} \quad \text{on } \partial\Omega\setminus\Gamma, \end{aligned} \tag{2.24}$$

Formally, as for equation (2.6) we obtain the weak formulation by multiplying (2.24) by a function $V\times Q := (\boldsymbol{v},q)\in[H_0^1(\Omega)]^2\times L_0^2(\Omega)$ and integrating by parts over $\Omega^* = \Omega_1\cup\Omega_2$, leading to the problem of finding $(\boldsymbol{u},p)\in V\times Q$ such that

$$\begin{aligned} (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v}))_{\Omega^*} - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v})_{\Omega^*} + (q,\boldsymbol{\nabla}\cdot\boldsymbol{u})_{\Omega^*} \\ - \int_\Gamma [\![p\,\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}]\!]\cdot\boldsymbol{v}\ \mathrm{d}s = (\boldsymbol{f},\boldsymbol{v})_\Omega, \quad \forall(\boldsymbol{v},q)\in V\times Q. \end{aligned}$$

Taking into account the following interface condition

$$[\![p\,\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u})\boldsymbol{n}]\!] = \sigma\kappa\boldsymbol{n} \quad \text{on } \Gamma, \tag{2.25}$$

which is identical to (1.10), we arrive at the following well posed formulation: Find $(\boldsymbol{u},p)\in V\times Q$ such that

$$B[(\boldsymbol{u},p),(\boldsymbol{v},q)] = (\boldsymbol{f},\boldsymbol{v})_\Omega + \int_\Gamma \sigma\kappa\boldsymbol{n}\cdot\boldsymbol{v}\ \mathrm{d}s, \quad \forall(\boldsymbol{v},q)\in V\times Q, \tag{2.26}$$

where

$$B[(\boldsymbol{u},p),(\boldsymbol{v},q)] = (2\mu\mathbf{D}(\boldsymbol{u}),\mathbf{D}(\boldsymbol{v}))_{\Omega^*} - (p,\boldsymbol{\nabla}\cdot\boldsymbol{v})_{\Omega^*} + (q,\boldsymbol{\nabla}\cdot\boldsymbol{u})_{\Omega^*}.$$

For the sake of readability, we omit in this section subscripts of normal vectors and jump operators, as their meaning is clear by the context of the domain of the integral they occur in.

2.5.2 The Finite Element Formulation

In a standard finite element method, the jump in the normal derivative of $\boldsymbol{u}$, resulting from the continuity of the flux when $\mu_1\neq\mu_2$, can be taken into account by letting Γ coincide with meshlines. However for two-fluid problems where the interface moves this implies remeshing for each timestep, at least close to the interface, or moving the mesh. The latter option presumes that the interface moves without changing its topology, like e.g. in fluid-structure interaction problems. Since in view of the unsteady problem we want to account for interface

topology changes, we cannot rely on moving the mesh. Instead we propose to solve (2.24) approximately using piecewise linear velocities and piecewise constant pressures on a family of conforming triangulations $\{\mathcal{T}_h\}_h$ of Ω which are *independent* of the location of the interface. To avoid loss in convergence order we relax the continuity requirement over Γ and allow the approximation to be discontinuous inside elements which intersect the interface.
We will use the following notation for mesh related quantities. Let h_K be the diameter of K and $h = \max_{K\in\mathcal{T}_h} h_K$. For any element K, let $K_i = K \cap \Omega_i$ denote the part of K in Ω_i. By $G_h := \{K \in \mathcal{T}_h : K \cap \Gamma \neq \emptyset\}$ we denote the set of elements that are intersected by the interface. For an element $K \in G_h$, let $\Gamma_K := \Gamma \cap K$ be the part of Γ in K.
We make the following assumptions regarding the mesh and the interface.

- A1: We assume that the triangulation is non-degenerate, i.e.,

$$h_K/\rho_K \leq C \quad \forall K \in \mathcal{T}_h$$

  where h_K is the diameter of K and ρ_K is the diameter of the largest ball contained in K and C is a given constant.

- A2: We assume that Γ intersects each element boundary ∂K exactly twice and each (open) edge at most once.

- A3: Let $\Gamma_{K,h}$ be the straight line segment connecting the points of intersection between Γ and ∂K (see Figure 2.1). We assume that Γ_K can be parametrized as a function of length on $\Gamma_{K,h}$; in local coordinates

$$\Gamma_{K,h} = \{(\xi,\eta) : 0 < \xi < |\Gamma_{K,h}|, \eta = 0\}$$

  and

$$\Gamma_K = \{(\xi,\eta) : 0 < \xi < |\Gamma_{K,h}|, \eta = \delta(\xi)\}.$$

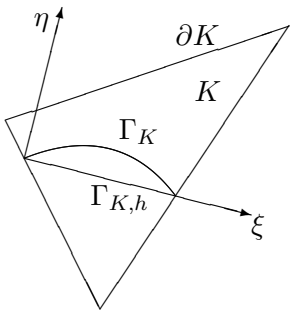


Figure 2.1: Local coordinates in K for Γ_K and $\Gamma_{K,h}$

Since we presume that the curvature of Γ is bounded the assumptions A2 and A3 are always fulfilled on sufficiently fine meshes. These assumptions essentially demand that *the interface is well resolved by the mesh.*
To obtain the finite element formulation corresponding to (2.26) we replace the test function $\boldsymbol{v}$ by $\tilde{\boldsymbol{v}}$ such that $\tilde{\boldsymbol{v}}|_{\Omega_i} = \boldsymbol{v}_i$ with $\boldsymbol{v}_i \in H^1(\Omega_i)$ and $\boldsymbol{v}_i = 0$ on the boundary $\partial\Omega$ thus allowing for a discontinuity over the interface Γ.

For the weighted averages across Γ of any sufficiently smooth function a discontinuous across Γ we will use the notation $\{a\} = \kappa_1 a|_{\Omega_1} + \kappa_2 a|_{\Omega_2}$ and $\langle a \rangle = \kappa_2 a|_{\Omega_1} + \kappa_1 a|_{\Omega_2}$ with $\kappa_i|_{\Gamma_K} = |K_i|/|K|$. Note that $\kappa_1 + \kappa_2 = 1$. We have

$$a_1 = \{a\} + \kappa_2 [\![a]\!] \text{ and } a_2 = \{a\} - \kappa_1 [\![a]\!],$$

and therefore

$$[\![ab]\!] = \{a\}[\![b]\!] + [\![a]\!]\langle b \rangle. \tag{2.27}$$

Integrating by parts and using (2.27) we obtain for the interface term

$$\int_\Gamma [\![(p\,\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}) \cdot \tilde{\boldsymbol{v}}]\!] \, \mathrm{d}s = \int_\Gamma \{p\,\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}\} \cdot [\![\tilde{\boldsymbol{v}}]\!] \, \mathrm{d}s + \int_\Gamma [\![p\,\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}]\!] \cdot \langle \tilde{\boldsymbol{v}} \rangle \, \mathrm{d}s. \tag{2.28}$$

Using now the interface condition (2.25) in the second term we have

$$\int_\Gamma [\![p\,\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}]\!] \cdot \langle \tilde{\boldsymbol{v}} \rangle \, \mathrm{d}s = \int_\Gamma \sigma\kappa \langle \tilde{\boldsymbol{v}} \cdot \boldsymbol{n} \rangle \, \mathrm{d}s. \tag{2.29}$$

After these preliminary considerations we are now ready to propose a finite element discretization of the problem. With this aim we introduce two conforming triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ such that

- For $i = 1, 2$, we have $\Omega_i \subset \bigcup_{K \in \mathcal{T}_i} K$.
- The union of $\mathcal{T}_1$ and $\mathcal{T}_2$ gives a conforming triangulation of all Ω.
- For every triangle $K \in \mathcal{T}_1 \cup \mathcal{T}_2$ we have $K \in \mathcal{T}_1 \cap \mathcal{T}_2 \Leftrightarrow K \cap \Gamma \neq \emptyset$.

See Figure 2.2 for an illustration of a one dimensional case.

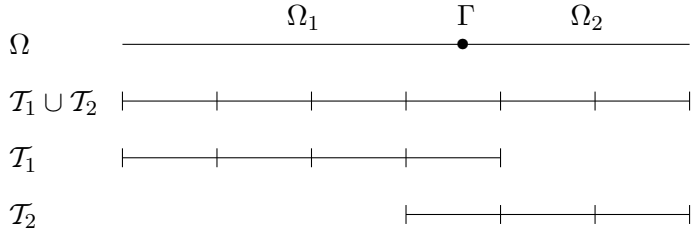


Figure 2.2: Example of triangulations $\mathcal{T}_1$ and $\mathcal{T}_2$ in a one dimensional case

Associated with $\mathcal{T}_1$ and $\mathcal{T}_2$ we have the finite element spaces

$$\begin{aligned}
V_{h,i} &= \{\boldsymbol{v} \in [C^0(\Omega_i)]^2 : \boldsymbol{v}|_K \in [\mathbb{P}_1(K)]^2 \ \forall K \in \mathcal{T}_i, \ \boldsymbol{v}|_{\partial\Omega} = \boldsymbol{0}\}, \\
V_h &= \{\boldsymbol{v} \in [L^2(\Omega)]^d : \boldsymbol{v}|_{\Omega_i} \in V_{h,i}, \ i = 1, 2\}, \\
Q_{h,i} &= \{q \in L^2_0(\Omega_i) : q|_K \in \mathbb{P}_0(K) \ \forall K \in \mathcal{T}_i\}, \\
Q_h &= \{q \in L^2(\Omega) : q|_{\Omega_i} \in Q_{h,i}, \ i = 1, 2\}.
\end{aligned}$$

The finite element discretization now takes the form:
Find $(\boldsymbol{u}_i, p_i) \in V_{h,i} \times Q_{h,i}$, $i = 1, 2$ such that $\boldsymbol{u}_h = (\boldsymbol{u}_1, \boldsymbol{u}_2)$, $p_h = (p_1, p_2)$ satisfy

$$B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{v}_h, q_h)] = (\boldsymbol{f}, \boldsymbol{v}_h) + \int_\Gamma \sigma\kappa\langle \boldsymbol{v}_h \cdot \boldsymbol{n}\rangle \, \mathrm{d}s + \sum_{K\in\mathcal{T}} \int_{\Gamma_K} \gamma_2 h_K \, \sigma\kappa \boldsymbol{n} \cdot [\![q_h \boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}]\!] \, \mathrm{d}s \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times Q_h \tag{2.30}$$

where

$$B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{v}_h, q_h)] = a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) + b_h(p_h, \boldsymbol{v}_h) - b_h(q_h, \boldsymbol{u}_h) + J(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h, q_h),$$

$$a_h(\boldsymbol{u}_h, \boldsymbol{v}_h) = \int_{\Omega^*} 2\mu \mathbf{D}(\boldsymbol{u}_h) : \mathbf{D}(\boldsymbol{v}_h) \mathrm{d}x - \int_\Gamma \{2\mu \mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\} \cdot [\![\boldsymbol{v}_h]\!] \, \mathrm{d}s - \int_\Gamma \{2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}\} \cdot [\![\boldsymbol{u}_h]\!] \, \mathrm{d}s + \int_\Gamma \gamma_1 h^{-1} [\![\boldsymbol{u}_h]\!] \cdot [\![\boldsymbol{v}_h]\!] \, \mathrm{d}s,$$

$$b_h(p_h, \boldsymbol{v}_h) = -\int_{\Omega^*} p_h \boldsymbol{\nabla} \cdot \boldsymbol{v}_h \, \mathrm{d}x + \int_\Gamma \{p_h\} [\![\boldsymbol{v}_h \cdot \boldsymbol{n}]\!] \, \mathrm{d}s,$$

and

$$J(\boldsymbol{u}_h, p_h, \boldsymbol{v}_h, q_h) = \sum_{K\in\mathcal{T}} \int_{\partial K \cup \Gamma_K} \gamma_2 h_K [\![p_h \boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}]\!] \cdot [\![q_h \boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}]\!] \mathrm{d}s.$$

The formulation (2.30) satisfies the following consistency relation

Lemma 2.5.1. *(Galerkin orthogonality) Let $(\boldsymbol{u}_h, p_h)$ be the solution of the finite element formulation (2.30) and assume that the solution of (2.26) $(\boldsymbol{u}, p)$ is in $[H^2(\Omega^*)]^2 \times H^1(\Omega^*)$. Then*

$$B_h[(\boldsymbol{u}_h - \boldsymbol{u}, p_h - p), (\boldsymbol{v}_h, q_h)] = 0 \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times Q_h.$$

Proof. Using the formulation (2.30) we may write

$$a_h(\boldsymbol{u}, \boldsymbol{v}_h) + b_h(p, \boldsymbol{v}_h) - b_h(q_h, \boldsymbol{u}) + J(\boldsymbol{u}, p, \boldsymbol{v}_h, q_h) = (\boldsymbol{f}, \boldsymbol{v}_h) + \int_\Gamma \sigma\kappa\langle \boldsymbol{v}_h \cdot \boldsymbol{n}\rangle \, \mathrm{d}s + \sum_{K\in\mathcal{T}} \int_{\Gamma_K} \gamma_2 h_K \, \sigma\kappa \boldsymbol{n} \cdot [\![q_h \boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}]\!] \, \mathrm{d}s \quad \forall (\boldsymbol{v}_h, q_h) \in V_h \times Q_h.$$

We use the regularity of $\boldsymbol{u}$ and p, as well as (2.25) to eliminate the third and forth term in a_h, and to obtain $J(\boldsymbol{u}, p, \boldsymbol{v}_h, q_h) = \sum_{K\in\mathcal{T}} \int_{\Gamma_K} \gamma_2 h_K \, \sigma\kappa \boldsymbol{n} \cdot [\![q_h \boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}]\!] \, \mathrm{d}s$. We note moreover that $b_h(q_h, \boldsymbol{u}) = 0$, and that the second term in a_h, together with the second term in $b_h(p, \boldsymbol{v}_h)$ and the second term on the right hand side, can be replaced by $\int_\Gamma [\![(p\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{u})\boldsymbol{n}) \cdot \boldsymbol{v}_h]\!] \, \mathrm{d}s$ on the left using (2.28) and (2.29). The claim now follows by integrating by parts in the other remaining two terms on the left hand side and using equation (2.24). □

A basis for V_h and Q_h is easily obtained from a standard finite element basis on the mesh by the introduction of new basis functions for the elements that intersect Γ. For V_h, the

standard piecewise linear interior nodal basis functions in Ω may be partitioned into the two sets $\{\psi_i^j\}_{j=1}^{N_i}$ of basis functions with support entirely in Ω_i, $i = 1, 2$, and the set $\{\psi_\Gamma^k\}_{k=1}^M$ of basis functions which are non-zero on Γ. For each of the latter, let

$$\psi_{\Gamma,i}^j(\boldsymbol{x}) := \begin{cases} \psi_\Gamma^j(\boldsymbol{x}) & \boldsymbol{x} \in \Omega_i \\ 0 & \boldsymbol{x} \notin \Omega_i \end{cases} .$$

Then $\bigcup_{i=1,2}(\{\psi_i^j\}_{j=1}^{N_i} \cup \{\psi_{\Gamma,i}^k\}_{k=1}^M)$ is a basis for V_h. An analogous procedure yields a basis for Q_h. As a consequence, the number of element shape functions on each element that intersects Γ is doubled.

2.5.3 Approximation Properties

We need to show that our approximating spaces V_h and Q_h have optimal approximation properties. This follows from some minor modifications of the analysis in [45]. We denote by C a generic positive constant independent of the mesh size h, and by ε a generic positive scalar in Young's inequality

$$ab \leq \frac{1}{2}\left(\frac{1}{\varepsilon}a^2 + \varepsilon b^2\right) \quad \forall a, b \in I\!R. \tag{2.31}$$

We remind that G_h denotes the set of elements that are intersected by the interface. We will use the following mesh dependent norms:

$$\|v\|_{1/2,h,\Gamma}^2 := \sum_{K \in G_h} h_K^{-1} \|v\|_{0,\Gamma_K}^2 ,$$

$$\|v\|_{-1/2,h,\Gamma}^2 := \sum_{K \in G_h} h_K^1 \|v\|_{0,\Gamma_K}^2 ,$$

and

$$|||(\boldsymbol{v}, q)|||^2 := \|\boldsymbol{v}\|_{0,\Omega}^2 + \|\boldsymbol{\nabla v}\|_{0,\Omega}^2 + \|\{2\mu \mathbf{D}(\boldsymbol{v})\boldsymbol{n}\}\|_{-1/2,h,\Gamma}^2 + \|[\![\boldsymbol{v}]\!]\|_{1/2,h,\Gamma}^2 + \|q\|_{0,\Omega}^2 + \sum_{K \in \mathcal{T}} \|[\![q\,\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v})\boldsymbol{n}]\!]\|_{-1/2,h,\partial K \cup \Gamma_K}^2 .$$

We note for future reference that

$$(u, v)_\Gamma \leq \|u\|_{1/2,h,\Gamma} \|v\|_{-1/2,h,\Gamma} . \tag{2.32}$$

To show that functions in the space (V_h, Q_h) approximate functions $(\boldsymbol{v}, q) \in [H_0^1(\Omega) \cap H^2(\Omega^*)]^2 \times [L_0^2(\Omega) \cap H^1(\Omega^*)]$ to the order h in the norm $|||\cdot|||$, we construct an interpolant of $(\boldsymbol{v}, q)$ by nodal interpolants of $[H^2]^2 \times H^1$ - extensions of $(\boldsymbol{v}_1, q_1)$ and $(\boldsymbol{v}_2, q_2)$ as follows. Choose extension operators $\mathsf{E}_i^k : H^k(\Omega_i) \to H^k(\Omega)$ such that $(\mathsf{E}_i^k w)|_{\Omega_i} = w$ and

$$\|\mathsf{E}_i^k w\|_{s,\Omega} \leq C \|w\|_{s,\Omega_i} , \quad \forall w \in H^s(\Omega_i),\ s = 0, \ldots, k. \tag{2.33}$$

Let I_h be the standard interpolant and C_h be the local L^2-projection on the piecewise constants. We define

$$(I_h^* \boldsymbol{v}, C_h^* q) := ((I_{h,1}^* \boldsymbol{v}_1, I_{h,2}^* \boldsymbol{v}_2), (C_{h,1}^* q_1, C_{h,2}^* q_2)), \tag{2.34}$$

where $I_{h,i}^* \boldsymbol{v}_i := (I_h \mathsf{E}_i^2 \boldsymbol{v}_i)|_{\Omega_i}$ and $C_{h,i}^* q_i := (C_h \mathsf{E}_i^1 q_i)|_{\Omega_i}$. The following theorem is valid.

Theorem 2.5.1. *Let (I_h^*, C_h^*) be a pair of interpolation operators defined as in (2.34). Then*

$$|||(\boldsymbol{v} - I_h^*\boldsymbol{v}, q - C_h^* q)||| \leq Ch(\|\boldsymbol{v}\|_{2,\Omega^*} + \|q\|_{1,\Omega^*}),$$
$$\forall (\boldsymbol{v}, q) \in [H_0^1(\Omega) \cap H^2(\Omega^*)]^2 \times L_0^2(\Omega) \cap H^1(\Omega^*).$$

For the proof of this theorem we need the following variant of a trace inequality on a reference element that we recall from [45] and state here without proof.

Lemma 2.5.2. *Map a triangle $K \in G_h$ onto the unit reference triangle $\tilde{K}$ by an affine map and denote by $\tilde{\Gamma}_{\tilde{K}}$ the corresponding image of Γ_K. Under the assumptions A1-A3 of Section 2.5.2 there exists a constant C, depending on Γ but independent of the mesh, such that*

$$\|w\|_{0,\tilde{\Gamma}_{\tilde{K}}}^2 \leq C\|w\|_{0,\tilde{K}}\|w\|_{1,\tilde{K}}, \quad \forall w \in H^1(\tilde{K}).$$

Proof. (Theorem 2.5.1)
Recall that $K_i = K \cap \Omega_i$ and let $\boldsymbol{v}_i^* = \mathsf{E}_i^2 \boldsymbol{v}_i$ denote the extension of $\boldsymbol{v}_i$ to Ω and similarly $q_i^* = \mathsf{E}_i^1 q_i$ denote the extension of q_i to Ω. By standard interpolation estimates for I_h and C_h respectively we now obtain

$$\|\boldsymbol{\nabla}(\boldsymbol{v}_i - I_{h,i}^*\boldsymbol{v}_i)\|_{0,K_i}^2 = \|\boldsymbol{\nabla}(\boldsymbol{v}_i^* - I_{h,i}^*\boldsymbol{v}_i^*)\|_{0,K_i}^2 \leq \|\boldsymbol{\nabla}(\boldsymbol{v}_i^* - I_{h,i}^*\boldsymbol{v}_i^*)\|_{0,K}^2 \leq Ch_K^2\|\boldsymbol{v}_i^*\|_{2,K}^2$$

and in the same fashion

$$\|q_i - C_{h,i}^* q_i\|_{0,K_i}^2 \leq Ch_K^2\|q_i^*\|_{1,K}^2$$

and

$$\|\boldsymbol{v}_i - I_{h,i}^*\boldsymbol{v}_i\|_{0,K_i}^2 \leq Ch_K^4\|\boldsymbol{v}_i^*\|_{2,K}^2.$$

Summing over all triangles we obtain using (2.33)

$$\|\boldsymbol{\nabla}(\boldsymbol{v}_i - I_{h,i}^*\boldsymbol{v}_i)\|_{0,\Omega_i}^2 \leq Ch^2 \sum_{K \cap \Omega_i \neq \emptyset} \|\boldsymbol{v}_i^*\|_{2,K}^2 \leq Ch^2\|\boldsymbol{v}_i\|_{2,\Omega_i}^2, \tag{2.35}$$

$$\|q_i - C_{h,i}^* q_i\|_{0,\Omega_i}^2 \leq Ch^2\|q_i\|_{1,\Omega_i}^2 \tag{2.36}$$

and

$$\|\boldsymbol{v}_i - I_{h,i}^*\boldsymbol{v}_i\|_{0,\Omega_i}^2 \leq Ch^4\|\boldsymbol{v}_i\|_{2,\Omega_i}^2. \tag{2.37}$$

We turn now to the jumps on the interface. Since the mesh is non-degenerate, it follows from Lemma 2.5.2, scaled by the map from the reference triangle, that

$$h_K^{-s}\|w\|_{0,\Gamma_K}^2 \leq C\Big(h_K^{-1-s}\|w\|_{0,K}^2 + h_K^{1-s}\|w\|_{1,K}^2\Big), \quad \forall w \in H^1(K).$$

Hence it follows, using once again standard interpolation estimates, that

$$\begin{aligned} h_K^{-1}\|[\![\boldsymbol{v} - I_h^*\boldsymbol{v}]\!]\|_{0,\Gamma_K}^2 &\leq Ch_K^{-1}\sum_i \|\boldsymbol{v}_i - I_{h,i}^*\boldsymbol{v}_i\|_{0,\Gamma_K}^2 = Ch_K^{-1}\sum_i \|\boldsymbol{v}_i^* - I_h\boldsymbol{v}_i^*\|_{0,\Gamma_K}^2 \\ &\leq C\sum_i \Big(h_K^{-2}\|\boldsymbol{v}_i^* - I_h\boldsymbol{v}_i^*\|_{0,K}^2 + \|\boldsymbol{v}_i^* - I_h\boldsymbol{v}_i^*\|_{1,K}^2\Big) \\ &\leq Ch_K^2\sum_i \|\boldsymbol{v}_i^*\|_{2,K}^2 \end{aligned}$$

and

$$\begin{aligned} h_K\|[\![q - C_h^* q]\!]\|_{0,\Gamma_K}^2 &\le Ch_K \sum_i \|q_i - C_{h,i}^* q_i\|_{0,\Gamma_K}^2 = Ch_K \sum_i \|q_i^* - C_h q_i^*\|_{0,\Gamma_K}^2 \\ &\le C \sum_i \Big(\|q_i^* - C_h q_i^*\|_{0,K}^2 + h_K^2 \|q_i^*\|_{1,K}^2 \Big) \\ &\le Ch_K^2 \sum_i \|q_i^*\|_{1,K}^2. \end{aligned}$$

Summing the contributions from $K \in G_h$, we get from (2.33) that

$$\|[\![\boldsymbol{v} - I_h^* \boldsymbol{v}]\!]\|_{1/2,h,\Gamma} \le Ch \sum_i \|\boldsymbol{v}_i^*\|_{2,\bigcup K \in G_h} \le Ch \|\boldsymbol{v}\|_{2,\Omega^*} \quad \text{and} \tag{2.38}$$

$$\|[\![q - C_h^* q]\!]\|_{-1/2,h,\Gamma} \le Ch \sum_i \|q_i^*\|_{1,\bigcup K \in G_h} \le Ch \|q\|_{1,\Omega^*}. \tag{2.39}$$

Next, we consider the mean viscous stress at the interface. Lemma 2.5.2 applied to $\boldsymbol{\nabla} w \cdot \boldsymbol{n}$ and scaling gives

$$h_K \|\boldsymbol{\nabla} w \cdot \boldsymbol{n}\|_{0,\Gamma_K}^2 \le C(\|w\|_{1,K}^2 + h_K^2 \|w\|_{2,K}^2), \quad \forall w \in H^2(K).$$

Using this result applied to $w = \boldsymbol{v}_i^* - I_h \boldsymbol{v}_i^*$ and again standard interpolation estimates, it follows that

$$\begin{aligned} h_K\|\{\boldsymbol{\nabla}(\boldsymbol{v} - I_h^* \boldsymbol{v})\boldsymbol{n}\}\|_{0,\Gamma_K}^2 &\le Ch_K \sum_i \|\boldsymbol{\nabla}(\boldsymbol{v}_i - I_{h,i}^* \boldsymbol{v}_i)\boldsymbol{n}\|_{0,\Gamma_K}^2 \\ &= Ch_K \sum_i \|\boldsymbol{\nabla}(\boldsymbol{v}_i^* - I_h \boldsymbol{v}_i^*)\boldsymbol{n}\|_{0,\Gamma_K}^2 \\ &\le C \sum_i \Big(h_K \|\boldsymbol{v}_i^* - I_h \boldsymbol{v}_i^*\|_{1,K}^2 + h_K^2 \|\boldsymbol{v}_i^* - I_h \boldsymbol{v}_i^*\|_{2,K}^2 \Big) \\ &\le Ch_K^2 \sum_i \|\boldsymbol{v}_i^*\|_{2,K}^2. \end{aligned}$$

Summing again the contributions from $K \in G_h$, we obtain from (2.33) that

$$\|\{\boldsymbol{\nabla}(\boldsymbol{v}_i - I_{h,i}^* \boldsymbol{v}_i)\boldsymbol{n}\}\|_{-1/2,h,\Gamma} \le Ch \|\boldsymbol{v}_i\|_{2,\Omega_i}. \tag{2.40}$$

In a further step, the jump of the stress over interface and element edges is split as follows:

$$\begin{aligned} &\sum_{K \in \mathcal{T}} \|[\![(q - C_h^* q)\boldsymbol{n} - 2\mu \mathbf{D}(\boldsymbol{v} - I_h^* \boldsymbol{v})\boldsymbol{n}]\!]\|_{-1/2,h,\partial K \cup \Gamma_K} \\ &\qquad \le \|[\![q - C_h^* q]\!]\|_{-1/2,h,\Gamma} + \|[\![2\mu \mathbf{D}(\boldsymbol{v} - I_h^* \boldsymbol{v})\boldsymbol{n}]\!]\|_{-1/2,h,\Gamma} \\ &\qquad\quad + \sum_{K \in \mathcal{T}} h_K \|[\![q - C_h^* q]\!]\|_{0,\partial K \setminus \Gamma} + \sum_{K \in \mathcal{T}} h_K \|[\![2\mu \mathbf{D}(\boldsymbol{v} - I_h^* \boldsymbol{v})\boldsymbol{n}]\!]\|_{0,\partial K \setminus \Gamma}. \end{aligned}$$

The first term is already bounded using (2.39) and the second term, the jump of the viscous stress at the interface, can be estimated like the average in (2.40), and we have thus

$$\|[\![q - C_h^* q]\!]\|_{-1/2,h,\Gamma} + \|[\![2\mu \mathbf{D}(\boldsymbol{v} - I_h^* \boldsymbol{v})\boldsymbol{n}]\!]\|_{-1/2,h,\Gamma} \le Ch(\|q\|_{1,\Omega^*} + \|\boldsymbol{v}\|_{2,\Omega^*}). \tag{2.41}$$

Finally, using the standard trace inequality for K_i, standard error estimates, and (2.33), the terms of jumps over element edges yield

$$\begin{aligned}
&\sum_{K\in\mathcal{T}} h_K \|[\![q - C_h^* q]\!]\|^2_{0,\partial K\setminus\Gamma} \\
&= \sum_{K\in\mathcal{T}} h_K \sum_i \|[\![q - C_h^* q]\!]\|^2_{0,\partial K\cap\Omega_i} \\
&= \sum_{K\in\mathcal{T}} h_K \sum_i \|[\![q_i^* - C_h q_i^*]\!]\|^2_{0,\partial K\cap\Omega_i} \\
&\leq C \sum_{K\in\mathcal{T}} h_K \sum_i \|q_i^*|_K - (C_h q_i^*)|_K\|^2_{0,\partial K\cap\Omega_i} \\
&\leq C \sum_{K\in\mathcal{T}} h_K \sum_i (h_{K_i}^{-1}\|q_i^* - C_h q_i^*\|^2_{0,K_i} + h_{K_i}\|q_i^* - C_h q_i^*\|^2_{1,K_i}) \\
&\leq C \sum_{K\in\mathcal{T}} h_K \sum_i (h_{K_i}\|q_i^*\|^2_{0,K_i} + h_{K_i}\|q_i^*\|^2_{1,K_i}) \\
&\leq C \sum_{K\in\mathcal{T}} h_K^2 \sum_i \|q_i\|^2_{1,K_i} \\
&\leq Ch^2\|q\|^2_{1,\Omega^*},
\end{aligned} \tag{2.42}$$

where we have used that $|C_h q_i^*|_{1,K_i} = 0$. Analogously, we obtain

$$\sum_{K\in\mathcal{T}} h_K \|[\![2\mu\mathbf{D}(\boldsymbol{v} - I_h^*\boldsymbol{v})\boldsymbol{n}]\!]\|^2_{0,\partial K\setminus\Gamma} \leq Ch^2\|\boldsymbol{v}\|_{2,\Omega^*}. \tag{2.43}$$

The claim now follows from equations (2.35), (2.36), (2.37), (2.38), (2.40), (2.41), (2.42) and (2.43). □

2.5.4 The inf-sup Condition

Since the Stokes problem is a saddle point problem, mere coercivity of the bilinear form is insufficient to prove optimal a priori error estimates. We need to show that the formulation satisfies an *inf-sup* condition with regard to the norm $|||\cdot|||$. We need the following trace inequality. The proof is generalized from a similar result in [45].

Lemma 2.5.3. *For any rank-two-tensor* $\mathbf{S}$ *for which* $\mathbf{S}|_{K_i} = \text{const}$, *the following trace inequalitiy holds:*

$$\|\{\mathbf{S}\,\boldsymbol{n}\}\|^2_{-1/2,h,\Gamma} \leq C_I\|\mathbf{S}\|^2_{0,\Omega^*}$$

Proof. Since on K_i $\mathbf{S}|_{K_i}$ is constant, we have

$$h_K\|\kappa_i\mathbf{S}_i\boldsymbol{n}\|^2_{0,\Gamma_K} \leq h_K\kappa_i^2|\Gamma_K||\mathbf{S}_i|^2 = h_K\kappa_i^2\frac{|\Gamma_K|}{|K_i|}\|\mathbf{S}_i\|^2_{0,K_i} = h_K\frac{|\Gamma_K||K_i|}{|K|^2}\|\mathbf{S}_i\|^2_{0,K_i} \leq C\|\mathbf{S}_i\|^2_{0,K_i}.$$

In the last step above we have used that $|\Gamma_K| \leq h_k$, $|K_i| \leq h_K^2$, and, since the mesh is non-degenerate, $|K| \geq ch_K^2$. The result follows by summation over the elements. □

Lemma 2.5.4. *For all functions* $\boldsymbol{v} \in V_h$, *the following Korn's inequality holds:*

$$\|\mathbf{D}(\boldsymbol{v})\|^2_{0,\Omega^*} + \|[\![\boldsymbol{v}]\!]\|^2_{1/2,h,\Gamma} \geq C_K(\|\boldsymbol{v}\|^2_{0,\Omega} + \|\boldsymbol{\nabla}\boldsymbol{v}\|^2_{0,\Omega^*} + \|[\![\boldsymbol{v}]\!]\|^2_{1/2,h,\Gamma}).$$

Proof. From [11] (eq. (1.19)), using $V_h \subset [H^1(\Omega^*)]^d$ and $\boldsymbol{v}|_{\partial\Omega} = 0$ for $\boldsymbol{v} \in V_h$, we have

$$\|\mathbf{D}(\boldsymbol{v})\|^2_{0,\Omega^*} + \|[\![\boldsymbol{v}]\!]\|^2_{1/2,h,\Gamma} \geq C|\boldsymbol{v}|^2_{1,\Omega^*} \quad \forall \boldsymbol{v} \in V_h,$$

while from [10], using $\boldsymbol{v}|_{\partial\Omega} = 0$, $\boldsymbol{v}|_{\Gamma_K} \in L^2(\Gamma_K)$ for $\boldsymbol{v} \in V_h$ and a standard error estimate, we have

$$|\boldsymbol{v}|^2_{1,\Omega^*} + \|[\![\boldsymbol{v}]\!]\|^2_{1/2,h,\Gamma} \geq C\|\boldsymbol{v}\|^2_{0,\Omega} \quad \forall \boldsymbol{v} \in V_h.$$

The result now follows immediately. □

Theorem 2.5.2. *Let $(\boldsymbol{u}_h, p_h) \in V_h \times Q_h$, $\gamma_1 > 4\mu_{\max}C_I > 0$ and $\gamma_2 > 0$, then*

$$c_s|||(\boldsymbol{u}_h, p_h)||| \leq \sup_{(\boldsymbol{v}_h,q_h)\in V_h\times Q_h} \frac{B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{v}_h, q_h)]}{|||(\boldsymbol{v}_h, q_h)|||},$$

for a suitable uniform constant $c_s > 0$.

Proof. We start by choosing $(\boldsymbol{v}_h, q_h) = (\boldsymbol{u}_h, p_h)$ to obtain

$$B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{u}_h, p_h)] = a_h(\boldsymbol{u}_h, \boldsymbol{u}_h) + J(\boldsymbol{u}_h, p_h, \boldsymbol{u}_h, p_h).$$

Applying now (2.32) and Lemma 2.5.3, we get

$$\begin{aligned}
B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{u}_h, p_h)] &= a_h(\boldsymbol{u}_h, \boldsymbol{u}_h) + J(\boldsymbol{u}_h, p_h, \boldsymbol{u}_h, p_h) \\
&\geq 2\|\mu^{1/2}\mathbf{D}(\boldsymbol{u}_h)\|^2_{0,\Omega^*} - 2\|\{2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\}\|_{-1/2,h,\Gamma}\|[\![\boldsymbol{u}_h]\!]\|_{1/2,h,\Gamma} \\
&\quad + \gamma_1\|[\![\boldsymbol{u}_h]\!]\|^2_{1/2,h,\Gamma} + J(\boldsymbol{u}_h, p_h, \boldsymbol{u}_h, p_h) \\
&\geq \mu_{\min}\|\mathbf{D}(\boldsymbol{u}_h)\|^2_{0,\Omega^*} + \left(\frac{1}{4\mu_{\max}C_I} - \frac{1}{\varepsilon}\right)\|\{2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\}\|^2_{-1/2,h,\Gamma} \\
&\quad + (\gamma_1 - \varepsilon)\|[\![\boldsymbol{u}_h]\!]\|^2_{1/2,h,\Gamma} + J(\boldsymbol{u}_h, p_h, \boldsymbol{u}_h, p_h)
\end{aligned}$$

with

$$\mu_{\max} = \max_{i=1,2}\mu_i \quad \text{and} \quad \mu_{\min} = \min_{i=1,2}\mu_i.$$

Taking ε such that $\gamma_1 > \varepsilon > 4\mu_{\max}C_I$, we obtain

$$\begin{aligned}
B_h[(\boldsymbol{u}_h, p_h), (\boldsymbol{u}_h, p_h)] \geq \quad & \mu_{\min}\|\mathbf{D}(\boldsymbol{u}_h)\|^2_{0,\Omega^*} + C_0^*\|\{2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\}\|^2_{-1/2,h,\Gamma} \\
& + C_0\|[\![\boldsymbol{u}_h]\!]\|^2_{1/2,h,\Gamma} + J(\boldsymbol{u}_h, p_h, \boldsymbol{u}_h, p_h),
\end{aligned} \tag{2.44}$$

with

$$C_0 = \gamma_1 - \varepsilon > 0 \qquad \text{and} \qquad C_0^* = \frac{1}{4\mu_{\max}C_I} - \frac{1}{\varepsilon} > 0.$$

By the surjectivity of the divergence operator from $[H^1_0(\Omega)]^2$ to $L^2_0(\Omega)$ there exists $\boldsymbol{v}_p \in [H^1_0(\Omega)]^2$ such that $-\boldsymbol{\nabla}\cdot\boldsymbol{v}_p = p_h$ and $\|\boldsymbol{v}_p\|_{1,\Omega} \leq c\|p_h\|_{0,\Omega}$. Taking now $(\boldsymbol{v}_h, q_h) = (I_h^C\boldsymbol{v}_p, 0)$ where I_h^C is the Clément interpolation operator on the space of piecewise affine *continuous* functions in V_h (note that this implies that $[\![I_h^C\boldsymbol{v}_p]\!]_\Gamma = 0$), we obtain

$$\begin{aligned}
B_h[(\boldsymbol{u}_h, p_h), (I_h^C\boldsymbol{v}_p, 0)] =& (2\mu\mathbf{D}(\boldsymbol{u}_h), \mathbf{D}(I_h^C\boldsymbol{v}_p)) - ([\![\boldsymbol{u}_h]\!], \{2\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}\})_\Gamma \\
& - \sum_{K\in\mathcal{T}}\int_{\partial K\cup\Gamma_K} \gamma_2 h_K [\![p_h\,\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}]\!]\cdot[\![2\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}]\!]\,\mathrm{d}s \\
& - (p_h, \boldsymbol{\nabla}\cdot(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)) + \|p_h\|^2_{0,\Omega}.
\end{aligned} \tag{2.45}$$

It follows that the integrand of the third term on the right hand side will be nonzero only on the element boundaries ∂K. Using Cauchy-Schwarz inequality, Young's inequality (2.31), the standard trace inequality, and the property $\|\boldsymbol{v}_p\|_{1,\Omega} \leq c\|p_h\|_{0,\Omega}$, we obtain that

$$\begin{aligned}
&\sum_{K\in\mathcal{T}} \int_{\partial K\cup\Gamma_K} \gamma_2 h_K [\![p_h\,\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}]\!]\cdot[\![2\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}]\!]\,\mathrm{d}s \\
&\qquad \leq \sum_{K\in\mathcal{T}} \left(\frac{\gamma_2}{\varepsilon} \|[\![p_h\,\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}]\!]\|^2_{-1/2,h,\partial K} + \varepsilon\gamma_2 \|[\![\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}]\!]\|^2_{-1/2,h,\partial K} \right) \\
&\qquad\qquad \leq \frac{1}{\varepsilon} J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h) + \varepsilon C_1\gamma_2\|p_h\|^2_{0,\Omega} \leq \frac{C_1\gamma_2}{\varepsilon} J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h) + \varepsilon\|p_h\|^2_{0,\Omega},
\end{aligned}$$

as

$$\begin{aligned}
\sum_{K\in\mathcal{T}} \|[\![\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}]\!]\|^2_{-1/2,h,\partial K} &\leq \mu^2_{\max} \sum_{K\in\mathcal{T}} h_K \|[\![\boldsymbol{\nabla}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}]\!]\|^2_{0,\partial K} \\
&\leq C\mu^2_{\max} \sum_{K\in\mathcal{T}} h_K \|\boldsymbol{\nabla}(I_h^C\boldsymbol{v}_p)\|^2_{0,\partial K} \\
&\leq C\mu^2_{\max} \sum_{K\in\mathcal{T}} C_t \|I_h^C\boldsymbol{v}_p\|^2_{2,K} \\
&= C\mu^2_{\max} \sum_{K\in\mathcal{T}} C_t \|I_h^C\boldsymbol{v}_p\|^2_{1,K} \\
&\leq C\mu^2_{\max} C_t \|I_h^C\boldsymbol{v}_p\|^2_{1,\Omega} \leq C\mu^2_{\max} C_t \|\boldsymbol{v}_p\|^2_{1,\Omega} \\
&\leq C\mu^2_{\max} C_t \|p_h\|^2_{0,\Omega} =: C_1\|p_h\|^2_{0,\Omega}.
\end{aligned} \tag{2.46}$$

Consider now the first and the fourth term in the right hand side of (2.45), where an integration by parts yields

$$\begin{aligned}
&(2\mu\mathbf{D}(\boldsymbol{u}_h), \mathbf{D}(I_h^C\boldsymbol{v}_p)) - (p_h, \boldsymbol{\nabla}\cdot(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)) \\
&\qquad = (2\mu\mathbf{D}(\boldsymbol{u}_h), \boldsymbol{\nabla}(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)) - (p_h, \boldsymbol{\nabla}\cdot(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)) + (2\mu\mathbf{D}(\boldsymbol{u}_h), \boldsymbol{\nabla}\boldsymbol{v}_p) \\
&\qquad\qquad = \sum_{K\in\mathcal{T}} \int_{\partial K\cup\Gamma_K} [\![2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n} - p_h\,\boldsymbol{n}]\!]\cdot(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)\,\mathrm{d}s + (2\mu\mathbf{D}(\boldsymbol{u}_h), \boldsymbol{\nabla}\boldsymbol{v}_p)
\end{aligned}$$

Clearly, by an application of (2.32) we have

$$\begin{aligned}
&(2\mu\mathbf{D}(\boldsymbol{u}_h), \mathbf{D}(I_h^C\boldsymbol{v}_p)) - (p_h, \boldsymbol{\nabla}\cdot(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)) \\
&\qquad \geq -\|[\![2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n} - p_h\,\boldsymbol{n}]\!]\|_{-1/2,h,\Gamma\cup(\bigcup_K \partial K)} \|(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)\cdot\boldsymbol{n}\|_{1/2,h,\Gamma\cup(\bigcup_K \partial K)} \\
&\qquad\qquad - 2\mu_{\max}\|\mathbf{D}(\boldsymbol{u}_h)\|_{0,\Omega^*}\ \|\boldsymbol{\nabla}\boldsymbol{v}_p\|_{0,\Omega}
\end{aligned}$$

using the trace inequality and the properties of $\boldsymbol{v}_p$. Applying the trace inequalities from Lemma 2.5.2

$$\|(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)\cdot\boldsymbol{n}\|^2_{1/2,h,\partial K} \leq C_t\|\boldsymbol{v}_p\|^2_{1,K}$$

and

$$\|(I_h^C\boldsymbol{v}_p - \boldsymbol{v}_p)\cdot\boldsymbol{n}\|^2_{1/2,h,\Gamma} \leq C_t \sum_{K\in G_h} \|\boldsymbol{v}_p\|^2_{1,K},$$

the properties of $\boldsymbol{v}_p$, and Young's inequality (2.31), we obtain

$$\begin{aligned}(2\mu\mathbf{D}(\boldsymbol{u}_h),\mathbf{D}(I_h^C\boldsymbol{v}_p)) - (p_h,\boldsymbol{\nabla}\cdot(I_h^C\boldsymbol{v}_p-\boldsymbol{v}_p))&\\ \geq -\frac{cC_t}{2\gamma_2\varepsilon}J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h) - \varepsilon\left\|p_h\right\|_{0,\Omega}^2 - \frac{\mu_{\max}^2 c}{\varepsilon}\left\|\mathbf{D}(\boldsymbol{u}_h)\right\|_{0,\Omega}^2 - \varepsilon\left\|p_h\right\|_{0,\Omega}^2&\\ \geq -\frac{C_2}{\gamma_2\varepsilon}J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h) - \frac{C_3}{\varepsilon}\left\|\mathbf{D}(\boldsymbol{u}_h)\right\|_{0,\Omega}^2 - 2\varepsilon\left\|p_h\right\|_{0,\Omega}^2.&\end{aligned}$$

For the second term in the right hand side of (2.45), we use (2.32), Young's inequality (2.31), Lemma 2.5.3 and the properties of $\boldsymbol{v}_p$ to obtain

$$\begin{aligned}-(\llbracket\boldsymbol{u}_h\rrbracket,\{2\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\boldsymbol{n}\})_\Gamma &\geq -\frac{C_I}{\varepsilon}\left\|\llbracket\boldsymbol{u}_h\rrbracket\right\|_{1/2,h,\Gamma}^2 - \varepsilon\left\|\mu\mathbf{D}(I_h^C\boldsymbol{v}_p)\right\|_{0,\Omega^*}\\ &\geq -\frac{C_4}{\varepsilon}\left\|\llbracket\boldsymbol{u}_h\rrbracket\right\|_{1/2,h,\Gamma}^2 - \varepsilon\left\|p_h\right\|_{0,\Omega}^2\end{aligned}$$

Collecting terms we may write

$$\begin{aligned}B_h[(\boldsymbol{u}_h,p_h),(I_h^C\boldsymbol{v}_p,0)] \geq (1-4\varepsilon)\left\|p_h\right\|_{0,\Omega}^2 - \frac{1}{\varepsilon}\Bigg(\left(C_1\gamma_2+\frac{C_2}{\gamma_2}\right)J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h)&\\ + C_3\|\mathbf{D}(\boldsymbol{u}_h)\|_{0,\Omega}^2 + C_4\|\llbracket\boldsymbol{u}_h\rrbracket\|_{1/2,h,\Gamma}^2\Bigg).&\end{aligned}$$

For the first term, we take $\varepsilon = 1/8$ to obtain

$$\begin{aligned}B_h[(\boldsymbol{u}_h,p_h),(I_h^C\boldsymbol{v}_p,0)] \geq \frac{1}{2}\|p_h\|_{0,\Omega}^2 - 8\big((C_1\gamma_2+C_2\gamma_2^{-1})J(\boldsymbol{u}_h,p_h,\boldsymbol{u}_h,p_h)&\\ + C_3\|\mathbf{D}(\boldsymbol{u}_h)\|_{0,\Omega}^2 + C_4\|\llbracket\boldsymbol{u}_h\rrbracket\|_{1/2,h,\Gamma}^2\big).&\end{aligned} \tag{2.47}$$

We take the linear combination of the test functions in (2.44) and (2.47), $(\boldsymbol{v}_h,q_h) = (\alpha\boldsymbol{u}_h + I_h^C\boldsymbol{v}_p, \alpha p_h)$, to obtain

$$\begin{aligned}B_h[(\boldsymbol{u}_h,p_h),(\boldsymbol{v}_h,q_h)] \;\geq\; &\frac{1}{2}\left\|p_h\right\|_{0,\Omega}^2\\ &+ (\alpha\mu_{\min} - 8C_3)\|\mathbf{D}(\boldsymbol{u}_h)\|_{0,\Omega^*}^2\\ &+ (\alpha C_0 - 8C_4)\|\llbracket\boldsymbol{u}_h\rrbracket\|_{1/2,h,\Gamma}^2\\ &+ \alpha C_0^*\|\{2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\}\|_{-1/2,h,\Gamma}^2\\ &+ (\alpha\gamma_2 - 8C_1\gamma_2^2 - 8C_2)\sum_{K\in\mathcal{T}}\|\llbracket p_h\boldsymbol{n} - 2\mu\mathbf{D}(\boldsymbol{u}_h)\boldsymbol{n}\rrbracket\|_{-1/2,h,\partial K\cup\Gamma_K}^2\end{aligned}$$

Now, fixing

$$\alpha = \max\left\{\frac{1}{\mu_{\min}}\left(\frac{1}{2C_K}+8C_3\right), \frac{1}{C_0}\left(\frac{1}{2C_K}+8C_4\right), \frac{1}{2C_0^*}, \frac{1}{2\gamma_2}+8(C_1\gamma_2+C_2\gamma_2^{-1})\right\}$$

and applying Lemma 2.5.4, we conclude that

$$\frac{1}{2}|||(\boldsymbol{u}_h,p_h)|||^2 \leq B_h[(\boldsymbol{u}_h,p_h),(\alpha\boldsymbol{u}_h+I_h^C\boldsymbol{v}_p,\alpha p_h)].$$

The claim now follows noting that there exists c_s such that $2c_s|||(\alpha\boldsymbol{u}_h + I_h^C\boldsymbol{v}_p,\alpha p_h)||| \leq |||(\boldsymbol{u}_h,p_h)|||$, which we prove in the following lemma. □

Lemma 2.5.5. *There exists c_s such that*

$$2c_s|||(\alpha \boldsymbol{u}_h + I_h^C \boldsymbol{v}_p, \alpha p_h)||| \leq |||(\boldsymbol{u}_h, p_h)|||.$$

Proof. Using the properties of $\boldsymbol{v}_p$, Lemma 2.5.3 and (2.46), we have

$$\begin{aligned}
2c_s|||(\alpha \boldsymbol{u}_h + I_h^C \boldsymbol{v}_p, \alpha p_h)||| &\leq 2c_s(|||(\alpha \boldsymbol{u}_h, \alpha p_h)||| + |||(I_h^C \boldsymbol{v}_p, 0)|||) \\
&\leq 2c_s \left(\alpha |||(\boldsymbol{u}_h, p_h)||| + \left(1 + 4C_I \mu_{\max}^2 + 4C_1\right)^{1/2} c \|p_h\|_{0,\Omega} \right) \\
&\leq 2c_s \left(\alpha + c \left(1 + 4C_I \mu_{\max}^2 + 4C_1\right)^{1/2} \right) |||(\boldsymbol{u}_h, p_h)|||
\end{aligned}$$

and hence the claim follows with

$$c_s = \frac{1}{2} \left(\alpha + c \left(1 + 4C_I \mu_{\max}^2 + 4C_1\right)^{1/2} \right)^{-1}.$$

□

It can be seen from the expressions for c_s and α that the presented analysis suggests a dependence of c_s on $\mu_{\max}/\mu_{\min}$ that is at least linear. A better theoretical result for the present scheme is not excluded, though unlikely.

2.5.5 A priori Error Estimate

Property 2.5.1. *Assume that the solution $(\boldsymbol{u}, p)$ to problem (2.26) resides in $[H^2(\Omega^*)]^d \times H^1(\Omega^*) \cap L_0^2(\Omega)$; then the finite element solution (2.30) satisfies the error estimate*

$$|||(\boldsymbol{u} - \boldsymbol{u}_h, p - p_h)||| \leq ch(\|\boldsymbol{u}\|_{2,\Omega^*} + \|p\|_{1,\Omega^*}).$$

Proof. In view of Theorem 2.5.1 we only need to show the inequality for $|||(\boldsymbol{u}_h - I_h^* \boldsymbol{u}, p_h - C_h^* p)|||$. By Theorem 2.5.2 and using Galerkin orthogonality we obtain

$$|||(\boldsymbol{u}_h - I_h^* \boldsymbol{u}, p_h - C_h^* p)||| \leq \frac{1}{c_s} \sup_{(\boldsymbol{v}_h, q_h) \in V_h \times Q_h} \frac{B_h[(\boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p), (\boldsymbol{v}_h, q_h)]}{|||(\boldsymbol{v}_h, q_h)|||}.$$

It remains to use interpolation estimates to bound the terms on the right hand side. Below, we will make repeated use of inequality (2.32), Theorem 2.5.1, and Lemma 2.5.2. Moreover, we will use the analogous version of inequality (2.39) for the average, as well as Lemma 2.5.3 and its simplification to scalars. Treating $B_h[(\boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p), (\boldsymbol{v}_h, q_h)]$ term wise we obtain

$$\begin{aligned}
a_h(\boldsymbol{u} - I_h^* \boldsymbol{u}, \boldsymbol{v}_h) &= (2\mu \mathbf{D}(\boldsymbol{u} - I_h^* \boldsymbol{u}), \mathbf{D}(\boldsymbol{v}_h))_{0,\Omega^*} - (\{2\mu \mathbf{D}(\boldsymbol{u} - I_h^* \boldsymbol{u})\boldsymbol{n}\}, [\![\boldsymbol{v}_h]\!])_{0,\Gamma} \\
&\quad - (\{2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}\}, [\![\boldsymbol{u} - I_h^* \boldsymbol{u}]\!])_{0,\Gamma} + (\gamma_1 [\![\boldsymbol{u} - I_h^* \boldsymbol{u}]\!], [\![\boldsymbol{v}_h]\!])_{1/2,h,\Gamma} \\
&\leq 2\mu_{\max} \|\mathbf{D}(\boldsymbol{u} - I_h^* \boldsymbol{u})\|_{0,\Omega^*} \|\mathbf{D}(\boldsymbol{v}_h)\|_{0,\Omega^*} \\
&\quad + \|\{2\mu \mathbf{D}(\boldsymbol{u} - I_h^* \boldsymbol{u})\boldsymbol{n}\}\|_{-1/2,h,\Gamma} \|[\![\boldsymbol{v}_h]\!]\|_{1/2,h,\Gamma} \\
&\quad + \|\{2\mu \mathbf{D}(\boldsymbol{v}_h)\boldsymbol{n}\}\|_{-1/2,h,\Gamma} \|[\![\boldsymbol{u} - I_h^* \boldsymbol{u}]\!]\|_{1/2,h,\Gamma} \\
&\quad + \gamma_1 \|[\![\boldsymbol{u} - I_h^* \boldsymbol{u}]\!]\|_{1/2,h,\Gamma} \|[\![\boldsymbol{v}_h]\!]\|_{1/2,h,\Gamma} \\
&\leq (2\mu_{\max} + 2 + \gamma_1) |||(\boldsymbol{u} - I_h^* \boldsymbol{u}, 0)||| \; |||(\boldsymbol{v}_h, 0)||| \\
&\leq ch \|\boldsymbol{u}\|_{2,\Omega^*} |||(\boldsymbol{v}_h, q_h)|||,
\end{aligned}$$

$$
\begin{aligned}
b_h(p - C_h^* p, \boldsymbol{v}_h) &= -(p - C_h^* p, \boldsymbol{\nabla} \cdot \boldsymbol{v}_h) + (\{p - C_h^* p\}, [\![\boldsymbol{v}_h]\!])_\Gamma \\
&\leq \|(p - C_h^* p)\|_{0,\Omega^*} \|\boldsymbol{\nabla} \boldsymbol{v}_h\|_{0,\Omega^*} \\
&\quad + \|\{p - C_h^* p\}\|_{-1/2,h,\Gamma} \|[\![\boldsymbol{v}_h]\!]\|_{1/2,h,\Gamma} \\
&\leq (|||(0, p - C_h^* p)||| + ch\|p\|_{1,\Omega^*}) \; |||(\boldsymbol{v}_h, 0)||| \\
&\leq ch\|p\|_{1,\Omega^*} |||(\boldsymbol{v}_h, q_h)|||,
\end{aligned}
$$

$$
\begin{aligned}
-b_h(q_h, \boldsymbol{u} - I_h^* \boldsymbol{u}) &= (q_h, \boldsymbol{\nabla} \cdot (\boldsymbol{u} - I_h^* \boldsymbol{u})) - (\{q_h\}, [\![(\boldsymbol{u} - I_h^* \boldsymbol{u}) \cdot \boldsymbol{n}]\!])_\Gamma \\
&\leq \|q_h\|_{0,\Omega^*} \|\boldsymbol{\nabla}(\boldsymbol{u} - I_h^* \boldsymbol{u})\|_{0,\Omega^*} \\
&\quad + \|\{q_h\}\|_{-1/2,h,\Gamma} \|[\![(\boldsymbol{u} - I_h^* \boldsymbol{u}) \cdot \boldsymbol{n}]\!]\|_{1/2,h,\Gamma} \\
&\leq (|||(0, q_h)||| + C_I \|q_h\|_{0,\Omega^*}) \; ch|||(\boldsymbol{u} - I_h^* \boldsymbol{u}, 0)||| \\
&\leq ch\|\boldsymbol{u}\|_{2,\Omega^*} |||(\boldsymbol{v}_h, q_h)|||,
\end{aligned}
$$

and

$$
\begin{aligned}
J(\boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p, \boldsymbol{v}_h, q_h) &\leq J(\boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p, \boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p)^{1/2} \\
&\quad J(\boldsymbol{v}_h, q_h, \boldsymbol{v}_h, q_h)^{1/2} \\
&\leq \gamma_2 |||(\boldsymbol{u} - I_h^* \boldsymbol{u}, p - C_h^* p)||| \; |||(\boldsymbol{v}_h, q_h)||| \\
&\leq ch(\|\boldsymbol{u}\|_{2,\Omega^*} + \|p\|_{1,\Omega^*}) \; |||(\boldsymbol{v}_h, q_h)|||.
\end{aligned}
$$

□

2.6 Interior Penalty Formulation of Advection Equation

Beside the flow equations (1.1)-(1.2), we also need to discretize the advection equation for the level set function (1.15).

2.6.1 Time Discretization and Weak Formulation

We first discretize equation (1.15) in time using the same second order backward differencing scheme (2.1) as for the flow equations, to obtain

$$
\begin{aligned}
\alpha \phi^{n+1} + \boldsymbol{\beta}^{n+1} \cdot \boldsymbol{\nabla} \phi^{n+1} &= f^{n+1} \quad \text{in } \Omega \\
\phi &= \phi_{in} \quad \text{on } \partial\Omega_{in}^{n+1}
\end{aligned}
\tag{2.48}
$$

with

$$
\begin{aligned}
\alpha &= \frac{3}{2\Delta t}, \\
\boldsymbol{\beta}^{n+1} &= \boldsymbol{u}^{n+1}, \\
f^{n+1} &= \frac{1}{2\Delta t}(4\phi^n - \phi^{n-1}) \quad \text{and} \\
\partial\Omega_{in}^{n+1} &= \{\boldsymbol{x} \in \partial\Omega : \boldsymbol{\beta}^{n+1}(\boldsymbol{x}) \cdot \boldsymbol{n} < 0\}.
\end{aligned}
$$

Again, we drop the superscripts indicating time levels for better readability. The semi-discretized level set advection equations take then the form of an advection-reaction problem:

$$
\begin{aligned}
\alpha \phi + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} \phi &= f \quad \text{in } \Omega, \\
\phi &= \phi_{in} \quad \text{on } \partial\Omega_{in}.
\end{aligned}
\tag{2.49}
$$

We multiply (2.49) by a function $\psi \in H^1(\Omega)$ and integrate by parts to find the weak formulation [91]: Find $\phi \in L^2(\Omega)$ such that

$$(\alpha\phi,\psi)-(\phi,\boldsymbol{\nabla}\cdot(\boldsymbol{\beta}\psi))+\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi,\psi\rangle_{\partial\Omega\setminus\partial\Omega_{in}}=(f,\psi)-\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi_{in},\psi\rangle_{\partial\Omega_{in}}\quad\forall\psi\in H^1(\Omega). \tag{2.50}$$

2.6.2 Interior Penalty Formulation

It is well known that the standard Galerkin method for advection-reaction problems is not stable if implemented without stabilization. We choose to apply a similar interior penalty scheme as for the flow equations, introduced in [18].
We first integrate equation (2.50) by parts again and obtain

$$(\alpha\phi,\psi)+(\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\phi,\psi)-\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi,\psi\rangle_{\partial\Omega_{in}}=(f,\psi)-\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi_{in},\psi\rangle_{\partial\Omega_{in}}$$

and introduce the stabilization term

$$j_\phi(\phi,\psi)=\langle\gamma_\phi h_f^2\,|\boldsymbol{\beta}\cdot\boldsymbol{n}|\,[\![\boldsymbol{\nabla}\phi]\!]_f,[\![\boldsymbol{\nabla}\psi]\!]_f\rangle_{\Gamma_I}.$$

Our finite element scheme for the advection-reaction equation then reads as follows:
Find $\phi_h \in V_h^k$ such that

$$c_h(\phi_h,\psi_h)=l_h(\psi_h)\quad\forall\psi_h\in V_h^k, \tag{2.51}$$

where

$$c_h(\phi,\psi)=(\alpha\phi,\psi)+(\boldsymbol{\beta}\cdot\boldsymbol{\nabla}\phi,\psi)-\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi,\psi\rangle_{\partial\Omega_{in}}+j_\phi(\phi,\psi), \tag{2.52}$$

$$l_h(\psi)=(f,\psi)-\langle\boldsymbol{\beta}\cdot\boldsymbol{n}\;\phi_{in},\psi\rangle_{\partial\Omega_{in}}. \tag{2.53}$$

2.6.3 Known Theoretical Results

In [18], Burman and Hansbo consider the advection-diffusion-reaction problem with possibly vanishing diffusion. Their scheme differs from ours only by a slightly different scaling of the penalty coefficient, and they choose the polynomial degree $k = 1$ fixed, i.e., piecewise linear elements. Supposing that the exact solution ϕ belongs to $H^2(\Omega)$, the following quasi-optimal a priori error estimate for the numerical solution ϕ_h holds:

$$\|\phi-\phi_h\|_{0,\Omega}+h^{1/2}\|\boldsymbol{\beta}\cdot\boldsymbol{\nabla}(\phi-\phi_h)\|_{0,\Omega}\leq Ch^{3/2}\,\|\phi\|_{2,\Omega}\,. \tag{2.54}$$

This estimate shows the loss of $h^{1/2}$ with respect to interpolation in the norm L^2, a result typical for stabilized methods. Note that the derivative of ϕ in streamline direction converges with optimal order.

Chapter 3

Algorithmic Aspects

In this chapter, we adress some algorithmic aspects of incompressible flow simulations with continuous interior penalty finite element methods in general, and of two fluid flow computations in particular. In Section 3.1, we present the iterative schemes we use to treat time dependence, coupling and nonlinearity of the flow equations. In Section 3.2, we give a short overview of linear algebraic solvers and present a general approach for the adaptive reuse of preconditioners in Subsection 3.2.3. In Section 3.3, we suggest solutions to reduce the two main drawbacks of continuous interior penalty methods: its costly assembly and its extended stencil. A simple but effective strategy of reducing the cost of the matrix assembly for two fluid flows is presented in Section 3.4. Finally, we recall in Section 3.5 different level set reinitialization methods and suggest a new combination of the interface local projection and a fast marching method in a new formulation.

3.1 Iterative Schemes

In this section, we present different iterative schemes for solving the two fluid model equations from Chapter 1. We choose to treat the most complicated case of the two fluid problems. The presentend schemes simplify trivially if problems with constant density and viscosity are considered.

3.1.1 Time Discretization

The time discretization of the Navier-Stokes equations is a subject of research on its own. Recent contributions include e.g. high order time integration schemes [61] and exponential integrators [78]. The increased cost of high order methods in time pays off if they are combined with high order methods in space. The latter require however sufficient spatial regularity of the exact solution in order to actually converge at the order they are designed for. In the case of two-fluid problems, regularity providing convergence orders beyond 2 cannot be expected, and other properties of the time discretization seem to be more important in this case.
A comparison [60] shows moreover that second order schemes can be competitive with respect to accuracy, depending on the problem at hand. They are competitive with respect to efficiency anyway.
We choose the second order backward differencing scheme (2.1) for several reasons:

- It is second order accurate.

- It is inconditionally stable.
- It can be extended to variable timesteps (see [3]).
- It does not need an initial pressure, because the spatial operators are evaluated only at the new time level. This property is shared with the time-continuous problem but not with most other time advancing schemes.
- Solutions of previous time levels, which have to be stored due to the multistep character of the time discretization, can be used for extrapolation, providing an initial guess for subsequent nonlinear iterations.

Let us recall the time discrete equations constituting our two fluid model, (2.2), (2.3) and (2.48):

$$\begin{aligned} &\frac{3\rho(\phi^{n+1})}{2\Delta t}\boldsymbol{u}^{n+1} + (\rho(\phi^{n+1})\hat{\boldsymbol{u}}^{n+1}\cdot\boldsymbol{\nabla})\boldsymbol{u}^{n+1} \\ &\qquad -\boldsymbol{\nabla}\cdot(2\mu(\phi^{n+1})\mathbf{D}(\boldsymbol{u}^{n+1})) + \boldsymbol{\nabla}p^{n+1} = \rho(\phi^{n+1})\left(\boldsymbol{g} + \frac{1}{2\Delta t}(4\boldsymbol{u}^n - \boldsymbol{u}^{n-1})\right), \\ &\qquad\qquad \boldsymbol{\nabla}\cdot\boldsymbol{u}^{n+1} = 0 \end{aligned} \tag{3.1}$$

and

$$\frac{3}{2\Delta t}\phi^{n+1} + \boldsymbol{u}^{n+1}\cdot\boldsymbol{\nabla}\phi^{n+1} = \frac{1}{2\Delta t}(4\phi^n - \phi^{n-1}), \tag{3.2}$$

where $\rho(\phi)$ and $\mu(\phi)$ are defined by (1.13) and (1.14).

3.1.2 Space Discretization - Coupled Problem

We apply the space discretizations: (2.13) to the Oseen problem (3.1) and (2.51) to the advection problem (3.2). In order to show all relevant couplings and parameter dependencies, we will change notation of the bilinear forms B_h (2.14) and c_h (2.52), and of the linear forms f_h (2.20) and l_h (2.53). We write

$$B_h[\alpha,\boldsymbol{\beta},\mu;(\boldsymbol{u},p),(\boldsymbol{v},q)], \qquad f_h(\boldsymbol{f},\phi,\mu,t;\boldsymbol{v}), \qquad c_h(\alpha,\boldsymbol{\beta};\phi,\psi) \qquad \text{and} \qquad l_h(f,\boldsymbol{\beta},t;\psi).$$

The meaning of the parameters is obvious from the original definitions of the forms, except maybe for the dependence of f_h on ϕ. It stems from the term $\langle\sigma\kappa\boldsymbol{n}_\Gamma,\boldsymbol{v}\rangle_\Gamma$, where the level set function ϕ determines the interface curvature κ by (1.20), the interface normal $\boldsymbol{n}_\Gamma$ by (1.18) and the integration domain Γ by (1.12). The linear forms f_h and l_h moreover depend on time t, because the functions $\boldsymbol{g}_D$, $\boldsymbol{g}_N$ and ϕ_{in} defining the boundary conditions may be time-dependent.
To keep down notation, we define furthermore

$$A_h[\phi,\hat{\boldsymbol{u}};(\boldsymbol{u},p),(\boldsymbol{v},q)] := B_h\left[\frac{3\rho(\phi)}{2\Delta t},\rho(\phi)\hat{\boldsymbol{u}},\mu(\phi);(\boldsymbol{u},p),(\boldsymbol{v},q)\right], \tag{3.3}$$

$$F_h(\phi,t;(\boldsymbol{v},q)) := f_h\left(\rho(\phi)\left(\boldsymbol{g} + \frac{1}{2\Delta t}(4\boldsymbol{u}^n - \boldsymbol{u}^{n-1})\right),\phi,\mu(\phi),t;\boldsymbol{v}\right), \tag{3.4}$$

$$C_h(\boldsymbol{\beta};\phi,\psi) := c_h\left(\frac{3}{2\Delta t},\boldsymbol{\beta};\phi,\psi\right) \text{ and}$$

$$L_h(\boldsymbol{\beta},t;\psi) := l_h\left(\frac{1}{2\Delta t}(4\phi^n - \phi^{n-1}),\boldsymbol{\beta},t;\psi\right).$$

The fully discrete coupled problem reads:
Given $\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$, ϕ^n and ϕ^{n-1}, find $(\boldsymbol{u}^{n+1}, p^{n+1}) \in W_h^k$ and $\phi^{n+1} \in V_h^k$ such that

$$A_h[\phi^{n+1}, \hat{\boldsymbol{u}}^{n+1}; (\boldsymbol{u}^{n+1}, p^{n+1}), (\boldsymbol{v}, q)] = F_h(\phi^{n+1}, t^{n+1}; (\boldsymbol{v}, q)) \quad \forall (\boldsymbol{v}, q) \in W_h^k, \tag{3.5}$$

$$C_h(\boldsymbol{u}^{n+1}; \phi^{n+1}, \psi) = L_h(\boldsymbol{u}^{n+1}, t^{n+1}; \psi) \qquad \forall \psi \in V_h^k. \tag{3.6}$$

To be fully consistent we should have $\hat{\boldsymbol{u}}^{n+1} = \boldsymbol{u}^{n+1}$, which would make problem (3.5) nonlinear.

3.1.3 Simple Splitting

In order to solve the coupled problem (3.5)-(3.6) approximately, one can replace it in a first step by the following simple splitting:
Given $\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$, ϕ^n and ϕ^{n-1}:

1. Find $(\boldsymbol{u}^{n+1}, p^{n+1}) \in W_h^k$ such that

$$A_h[\hat{\phi}^{n+1}, \hat{\boldsymbol{u}}^{n+1}; (\boldsymbol{u}^{n+1}, p^{n+1}), (\boldsymbol{v}, q)] = F_h(\hat{\phi}^{n+1}, t^{n+1}; (\boldsymbol{v}, q)) \quad \forall (\boldsymbol{v}, q) \in W_h^k, \tag{3.7}$$

   with $\hat{\boldsymbol{u}}^{n+1} = 2\boldsymbol{u}^n - \boldsymbol{u}^{n-1}$ and $\hat{\phi}^{n+1} = 2\phi^n - \phi^{n-1}$.

2. Find $\phi^{n+1} \in V_h^k$ such that

$$C_h(\boldsymbol{u}^{n+1}; \phi^{n+1}, \psi) = L_h(\boldsymbol{u}^{n+1}, t^{n+1}; \psi) \qquad \forall \psi \in V_h^k. \tag{3.8}$$

In (3.7), the approximate level set function $\hat{\phi}^{n+1}$ and the approximate velocity $\hat{\boldsymbol{u}}^{n+1}$ are obtained by extrapolation, using values on previous time levels which have to be stored for the right hand side anyway. These explicit approximations allow to solve for $\boldsymbol{u}^{n+1}$ which can then be used to solve for ϕ^{n+1} in (3.8).
This splitting formally conserves second order accuracy, but because of the explicite character, it violates the unconditional stability of the time discretization.

3.1.4 Fixpoint Formulation

In order to reduce splitting errors introduced by the scheme (3.7)-(3.8), we consider the following fixpoint formulation:
Given $\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$, ϕ^n and ϕ^{n-1}:

1. Set $i = 0$, $\boldsymbol{u}_i^{n+1} = 2\boldsymbol{u}^n - \boldsymbol{u}^{n-1}$ and $\phi_i^{n+1} = 2\phi^n - \phi^{n-1}$.

2. Find $(\boldsymbol{u}_{i+1}^{n+1}, p_{i+1}^{n+1}) \in W_h^k$ such that

$$A_h[\phi_i^{n+1}, \boldsymbol{u}_i^{n+1}; (\boldsymbol{u}_{i+1}^{n+1}, p_{i+1}^{n+1}), (\boldsymbol{v}, q)] = F_h(\phi_i^{n+1}, t^{n+1}; (\boldsymbol{v}, q)) \quad \forall (\boldsymbol{v}, q) \in W_h^k.$$

3. Find $\phi_{i+1}^{n+1} \in V_h^k$ such that

$$C_h(\boldsymbol{u}_{i+1}^{n+1}; \phi_{i+1}^{n+1}, \psi) = L_h(\boldsymbol{u}_{i+1}^{n+1}, t^{n+1}; \psi) \qquad \forall \psi \in V_h^k.$$

4. If $i + 1 < i_{\max}$ or $d[(\boldsymbol{u}_{i+1}^{n+1}, p_{i+1}^{n+1}, \phi_{i+1}^{n+1}), (\boldsymbol{u}_i^{n+1}, p_i^{n+1}, \phi_i^{n+1})] > \varepsilon$,
   then increment i: $i \leftarrow i + 1$ and go to 2.

5. Accept $\boldsymbol{u}^{n+1} = \boldsymbol{u}_{i+1}^{n+1}$, $p^{n+1} = p_{i+1}^{n+1}$ and $\phi^{n+1} = \phi_{i+1}^{n+1}$.

where $i_{\max}$ denotes the maximal number of fixpoint iterations and ε the fixpoint tolerance. In step 4., the function d denotes a suitable measure for the increment of the solution. We use the following distance measure:

$$d[(\boldsymbol{u}, p, \phi), (\tilde{\boldsymbol{u}}, \tilde{p}, \tilde{\phi})] = \left(\frac{\|\boldsymbol{u} - \tilde{\boldsymbol{u}}\|_{0,\Omega}^2}{\|\boldsymbol{u}\|_{0,\Omega}^2} + \frac{\|p - \tilde{p}\|_{0,\Omega}^2}{\|p\|_{0,\Omega}^2} + \frac{\|\phi - \tilde{\phi}\|_{0,\Omega}^2}{\|\phi\|_{0,\Omega}^2} \right)^{1/2}. \tag{3.9}$$

Note that:

- By setting $i_{\max} = 1$, we recover the simple splitting (3.7)-(3.8).
- The fixpoint tolerance ε has to be chosen as $\mathcal{O}(h^m)$ if order m convergence has to be assured. If the finite element scheme is optimal with respect to interpolation, this means $m = k + 1$.

3.1.5 Aitken Formulation

Under certain circumstances, the fixpoint formulation may not converge. In this case, one possible remedy is to introduce a relaxation factor. Following [29], we decide to apply a multi-dimensional version of the Aitken relaxation. This gives us the following Aitken formulation: Given $\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$, ϕ^n and ϕ^{n-1}:

1. Set $i = 0$, $\boldsymbol{u}_i^{n+1} = 2\boldsymbol{u}^n - \boldsymbol{u}^{n-1}$ and $\phi_i^{n+1} = 2\phi^n - \phi^{n-1}$.
2. Find $(\tilde{\boldsymbol{u}}_{i+1}^{n+1}, \tilde{p}_{i+1}^{n+1}) \in W_h^k$ such that
$$A_h[\phi_i^{n+1}, \boldsymbol{u}_i^{n+1}; (\tilde{\boldsymbol{u}}_{i+1}^{n+1}, \tilde{p}_{i+1}^{n+1}), (\boldsymbol{v}, q)] = F_h(\phi_i^{n+1}, t^{n+1}; (\boldsymbol{v}, q)) \quad \forall (\boldsymbol{v}, q) \in W_h^k.$$
3. Find $\tilde{\phi}_{i+1}^{n+1} \in V_h^k$ such that
$$C_h(\tilde{\boldsymbol{u}}_{i+1}^{n+1}; \tilde{\phi}_{i+1}^{n+1}, \psi) = L_h(\tilde{\boldsymbol{u}}_{i+1}^{n+1}, t^{n+1}; \psi) \qquad \forall \psi \in V_h^k.$$
4. Compute the residuals
$$\bar{\boldsymbol{u}}_i^{n+1} = \boldsymbol{u}_i^{n+1} - \tilde{\boldsymbol{u}}_{i+1}^{n+1}, \qquad \bar{p}_i^{n+1} = p_i^{n+1} - \tilde{p}_{i+1}^{n+1}, \qquad \bar{\phi}_i^{n+1} = \phi_i^{n+1} - \tilde{\phi}_{i+1}^{n+1}.$$
5. If $i > 0$, compute the increments
$$\begin{aligned} \delta\boldsymbol{u} &= \boldsymbol{u}_i^{n+1} - \boldsymbol{u}_{i-1}^{n+1}, & \delta p &= p_i^{n+1} - p_{i-1}^{n+1}, & \delta\phi &= \phi_i^{n+1} - \phi_{i-1}^{n+1} \\ \delta\bar{\boldsymbol{u}} &= \bar{\boldsymbol{u}}_i^{n+1} - \bar{\boldsymbol{u}}_{i-1}^{n+1}, & \delta\bar{p} &= \bar{p}_i^{n+1} - \bar{p}_{i-1}^{n+1}, & \delta\bar{\phi} &= \bar{\phi}_i^{n+1} - \bar{\phi}_{i-1}^{n+1} \end{aligned}$$
and the relaxation factor
$$\omega = \frac{(\delta\bar{\boldsymbol{u}}, \delta\boldsymbol{u}) + (\delta\bar{p}, \delta p) + (\delta\bar{\phi}, \delta\phi)}{(\delta\bar{\boldsymbol{u}}, \delta\bar{\boldsymbol{u}}) + (\delta\bar{p}, \delta\bar{p}) + (\delta\bar{\phi}, \delta\bar{\phi})},$$
else set $\omega = \omega_0$.

6. Compute the new iterates

$$\boldsymbol{u}_{i+1}^{n+1} = \boldsymbol{u}_i^{n+1} - \omega \bar{\boldsymbol{u}}_{i+1}^{n+1}, \qquad p_{i+1}^{n+1} = p_i^{n+1} - \omega \bar{p}_{i+1}^{n+1}, \qquad \phi_{i+1}^{n+1} = \phi_i^{n+1} - \omega \bar{\phi}_{i+1}^{n+1}.$$

7. If $i+1 < i_{\max}$ or $d[(\boldsymbol{u}_{i+1}^{n+1}, p_{i+1}^{n+1}, \phi_{i+1}^{n+1}), (\boldsymbol{u}_i^{n+1}, p_i^{n+1}, \phi_i^{n+1})] > \varepsilon\,\omega$, then increment i: $i \leftarrow i+1$ and go to 2.

8. Accept $\boldsymbol{u}^{n+1} = \boldsymbol{u}_{i+1}^{n+1}$, $p^{n+1} = p_{i+1}^{n+1}$ and $\phi^{n+1} = \phi_{i+1}^{n+1}$,

where $i_{\max}$ denotes the maximal number of Aitken iterations, ε the tolerance criterion for convergence and ω_0 the default initial relaxation factor. In step 7., we use the same distance measure (3.9) as for the fixpoint formulation.
Remarks:

- The Aitken formulation with $\omega = 1$ corresponds to the fixpoint formulation.
- By setting $i_{\max} = 1$, we recover the simple splitting (3.7)-(3.8).
- The tolerance criterion ε has again to be chosen as $\mathcal{O}(h^m)$ if order m convergence has to be assured.
- The initial relaxation factor ω_0 is usually chosen quite small, which means to be cautious in the beginning and take a small increment.
- In step 5., one has to assure that the denominator is nonzero, otherwise a default value (e.g. ω_0) is applied. It is also common practice to allow only values of ω within certain bounds, i.e., to set ω to some upper or lower default value if the computation gives a value above or below the upper or lower bound, respectively.
- In step 7., we multiply the tolerance ε by ω. This amounts to requiring the residuals to be small, and not only the increments.

3.2 Linear Algebraic Solvers

In this Section, we give a short overview of state of the art methods for solving the linear systems stemming from discretizations of incompressible fluid flow equations. Then, we present a general adaptive algorithm for reusing incomplete factorization preconditioners in the context of sequences of matrices with slowly varying spectra.

3.2.1 Problem Setting

In the course of the iterative schemes presented in Section 3.1, one needs to solve discrete linear variational problems for $(\boldsymbol{u}, p) \in W_h^k$ and for $\phi \in V_h^k$ of the form

$$A_h[\phi, \hat{\boldsymbol{u}}; (\boldsymbol{u}, p), (\boldsymbol{v}, q)] = F_h(\phi; (\boldsymbol{v}, q)) \quad \forall (\boldsymbol{v}, q) \in W_h^k,$$
$$C_h(\boldsymbol{\beta}; \phi, \psi) = L_h(\boldsymbol{\beta}; \psi) \quad \forall \psi \in V_h^k.$$

By introducing suitable bases to the discrete spaces such as

$$W_h^k = \mathrm{span}_{i=1\ldots N}\{\varphi_i\} \qquad \text{and} \qquad V_h^k = \mathrm{span}_{i=1\ldots M}\{\theta_i\},$$

these problems can be rewritten in algebraic form

$$\begin{aligned} \boldsymbol{V}^\top \mathbf{A}(\phi,\hat{\boldsymbol{u}})\boldsymbol{U} = \boldsymbol{V}^\top \boldsymbol{F}(\phi) \quad \forall \boldsymbol{V} \in I\!R^N \quad &\Rightarrow \quad \mathbf{A}(\phi,\hat{\boldsymbol{u}})\boldsymbol{U} = \boldsymbol{F}(\phi) \\ \boldsymbol{Y}^\top \mathbf{C}(\boldsymbol{\beta})\boldsymbol{X} = \boldsymbol{Y}^\top \boldsymbol{L}(\boldsymbol{\beta}) \quad \forall \boldsymbol{X} \in I\!R^M \quad &\Rightarrow \quad \mathbf{C}(\boldsymbol{\beta})\boldsymbol{X} = \boldsymbol{L}(\boldsymbol{\beta}) \end{aligned}$$

where

$$(\boldsymbol{u},p) = \sum_{i=1}^{N} (\boldsymbol{U})_i \; \varphi_i, \qquad (\mathbf{A}(\phi,\hat{\boldsymbol{u}}))_{ij} = A_h[\phi,\hat{\boldsymbol{u}};\varphi_j,\varphi_i], \qquad (\boldsymbol{F}(\phi))_i = F_h(\phi;\varphi_i),$$

$$\phi = \sum_{i=1}^{M} (\boldsymbol{X})_i \; \theta_i, \qquad (\mathbf{C}(\boldsymbol{\beta}))_{ij} = C_h(\boldsymbol{\beta};\theta_j,\theta_i), \qquad (\boldsymbol{L}(\boldsymbol{\beta}))_i = L_h(\boldsymbol{\beta};\theta_i).$$

Because N and M are big, the matrices $\mathbf{A} \in I\!R^{N\times N}$ and $\mathbf{C} \in I\!R^{M\times M}$ are of considerable size. Thanks to the local support of the finite element bases, the matrices are sparse however.
The linear systems with the matrix $\mathbf{C}$ associated to the advection-reaction problem can be solved without major concerns using standard iterative methods like the generalized minimial residual method (GMRes) preconditioned by incomplete LU factorizations (ILU).
For linear systems originating from a discretization of the Stokes, Oseen or Navier-Stokes equations, standard iterative methods are usually not applicable or insatisfactory, because the matrix $\mathbf{A}$ is not positive definite, and the system represents a saddle point problem. The development of more appropriate methods is a subject of ongoing research.

3.2.2 State of the Art Methods for Saddle Point Problems

We give here a short overview of state of the art methods for saddle point problems emanating from incompressible fluid flow. See [4] for an overview of solution methods for saddle point problems in general.
Operator splitting methods (see e.g. [41]) are one big class of methods addressing the issue of the solution of the linear systems from incompressible fluid flow problems. They are also known as fractional step methods, projection methods or pressure correction metods, because the incompressibility constraint is imposed in a second or third step (after advection and diffusion), projecting the velocity on a (weakly) divergence free space and correcting the pressure acordingly. The different steps are separated on a continuous level and discretized and solved separately. The monograph [86] gives a broad overview of projection methods, and a detailed analysis can be found e.g. in [43]. Truly consistent splittings have been introduced and analyzed in [44].
A related approach applies the splitting on an the algebraic level (see e.g. [90]), leading to the *algebraic factorization methods*. The matrix $\mathbf{A}$ is split into four blocks, separating degrees of freedom of velocity and pressure in both rows and columns:

$$\mathbf{A} = \begin{pmatrix} \mathbf{D} & \mathbf{B} \\ -\mathbf{B}^\top & \mathbf{J} \end{pmatrix},$$

where from (2.14) we can see that for our discretization the submatrix $\mathbf{D}$ is associated to the sum of bilinear forms $a_h + j_{\boldsymbol{\beta}} + j_{\text{div}}$, (2.15)+(2.17)+(2.18), $\mathbf{B}$ is associated to the bilinear form b_h, (2.16), and $\mathbf{J}$ is associated to the bilinear form j_p, (2.19). The basic idea is a block LU factorization allowing to solve for the pressure and the velocity separately.

Yet another class of methods are the *multigrid methods*. They use a hierarchy of grids that allow to use a computationally inexpensive coarse grid solution projected on a finer grid as an initial value for the fine grid solver. A robust multigrid solver for the Navier-Stokes problem in rotation form has been introduced and analyzed in [80]. A comparison of linear and nonlinear multigrid methods for the Navier-Stokes equations can be found in [74], and [117] provides a comparison of multigrid methods for the Navier-Stokes equations with standard Krylov methods and combinations of the two. Multigrid methods for discretizations of the Navier-Stokes equations using interior penalty terms have been studied in [83].
As for the splitting methods, there is also an algebraic variant of the multigrid method. *Algebraic multigrid methods* try to select subsets of matrix indices to be combined together as the fine grid cells are combined together to coarser ones in multigrid methods. The key issue is to extract the connectivity and dependence relations from the matrix only. Algebraic multigrid methods for the velocity-pressure coupled matrix of a discretized Oseen problem have been studied in [115].
These methods can be combined and/or used as preconditioners for more standard iterative solvers. Recent developments of *preconditioners* include block preconditioners for the steady [35] and the unsteady [34] linearized Navier-Stokes equations, block factorized preconditioners for high order in time accuracy [114], block triangular preconditioners for stabilized mixed finite element discretizations [24] and preconditioners for the Uzawa algorithm applied to stabilized mixed finite element matrices [25]. A comparison of parallel block multigrid preconditioners can be found in [97]. Standard incomplete LU factorizations as preconditioners have been compared in [113] to band block LU factorizations for the steady two dimensional Navier-Stokes equations.
In the present work, we choose to use the generalized minimial residual method (GMRes) preconditioned by a threshold incomplete LU factorizations (ILUT), applied to the full matrix $\mathbf{A}$. This choice is motivated by the following reasons:

- In inf-sup stable methods without stabilization, the lower right submatrix $\mathbf{J}$ of $\mathbf{A}$ is zero. Standard preconditioning techniques like incomplete LU factorizations need to be applied in a pivoting version (e.g. ILUTP) in this case, which is computationally more expensive in terms of both memory and computing time. Since we are using a stabilized finite element method, $\mathbf{J}$ is nonzero and positive semidefinite. For $p_h \in V_h^1$, i.e., piecewise linear pressures, the kernel of $j_p(\cdot, q)$ is the set of globally affine pressures. So the kernel of $\mathbf{J}$ has dimension $d + 1$ in this case, independent of h. Incomplete factorizations can cope well with small kernels, so we are not concerned by this issue. The situation might be different for higher order pressure approximations though.

- Splitting methods are often based on approximations which introduce artificial boundary conditions on the pressure causing artificial boundary layers.

- The currently available computing power allows to solve more and more complex problems within reasonable time even without applying optimally effective algorithms, at least in two dimensions.

- The focus of the present work is not on linear algebraic solution techniques for saddle point problems.

We use an effective yet general approach to reuse the ILUT preconditioners in an adaptive manner, reducing thereby the computation times considerably. This method is presented in

the following section.

3.2.3 A Preconditioning Strategy with Adaptively Reusable Preconditioner

In this thesis, we have found particularly interesting to develop and apply a preconditioning strategy with automatic reuse of the preconditioner. The algorithm is based on the following considerations: In the course of the iterative schemes presented in Section 3.1, one needs to solve linear systems with sparse matrices of considerable size repeatedly. This holds for the Oseen problem as well as for the level set advection problem. From one solve to the next one, the matrices change, but as the problems are physically related, the spectra of the matrices remain similar and a preconditioner created for the previous solve can still serve for the current solve. The difficulty lies in determining how long a preconditioner can be reused and when it should be replaced by a new one.
We propose here a scheme which takes this decision adaptively. Consider the general case where one has solved a sequence of N linear systems $\mathbf{A}_i\boldsymbol{x}_i = \boldsymbol{b}_i$, $i = 1, 2, \ldots N$ with the same preconditioner and one now has to solve the next system for $i = N + 1$. Denote by t_i the CPU time spent for solving the i-th system, and by t_{PC} the CPU time spent for constructing the preconditioner. The decision to be taken at this stage is whether or not to reuse the preconditioner for $i = N + 1$.
If the preconditioner is not reused, then its cost has to be amortized over the solves $i = 1$ to N. The average cost is then:

$$T_N = \frac{1}{N}\left(t_{PC} + \sum_{i=1}^{N} t_i\right)$$

If the preconditioner is reused, then its cost can be amortized over (at least) $N + 1$ solves $i = 1 \ldots N + 1$. The average cost T_N is then:

$$T_{N+1} = \frac{1}{N+1}\left(t_{PC} + \sum_{i=1}^{N+1} t_i\right)$$

Note that t_{N+1} is not known at the given stage and has to be replaced by some estimate $\hat{t}_{N+1}$ yet to be defined. The estimated average cost $\hat{T}_{N+1}$ is thus:

$$\hat{T}_{N+1} = \frac{1}{N+1}\left(t_{PC} + \hat{t}_{N+1} + \sum_{i=1}^{N} t_i\right)$$

If we suppose that when rebuilding the preconditioner, the next sequence of systems will allow to be solved with the same average cost, then rebuilding should be done if and only if $\hat{T}_{N+1} > T_N$ which can be transformed to the following criterion:

$$\hat{t}_{N+1} > \frac{1}{N}\left(t_{PC} + \sum_{i=1}^{N} t_i\right) = T_N \tag{3.10}$$

As the convergence properties of the sequence of systems cannot be easily estimated a priori, extrapolation of costs should be of lowest possible order, and the estimate $\hat{t}_{N+1} = t_N$ is usually preferred. Thus the preconditioner is rebuilt if and only if $t_N > T_N$.
In order to depend less on possibly incertain measurements of CPU time t, the algorithm can also work on iteration counts c. A fictitious iteration count c_{PC} then has to be assigned to

the cost of constructing the preconditioner, typically $c_{PC} = c_1\ t_{PC}/t_1$. CPU timing is then needed only during preconditioner recomputation and during the first solve following this recomputation. All formulas above hold then analogously with times t replaced by iteration counts c. Using moreover the estimator $\hat{c}_{N+1} = c_N$, the rebuild criterion (3.10) becomes

$$c_N > \frac{1}{N}\left(c_{PC} + \sum_{i=1}^{N} c_i\right) = C_N$$

Working with iteration counts has another advantage: C_N gives us an upper bound of the iteration count, beyond which it is not efficient to continue to reuse the preconditioner. It is thus natural to use C_N as maximum iteration count parameter for the linear solver when reusing the preconditioner. Should the solver not converge within C_N iterations, it is worth paying the price and rebuilding the preconditioner, in order to find the solution in a subsequent solve with the new preconditioner and using the final approximation of the first solving tentative as initial guess. This allows to avoid iterating too long with a preconditioner not suitable for reuse any more.

3.3 Treatment of Stabilization Terms

Continuous interior penalty stabilized methods have two main drawbacks:

- The assembly of the stabilization matrix is costly because the gradient jump terms require the evaluation of gradients on two elements and therefore the update of two geometric transformations for each internal face to be integrated on. Moreover, the weights in these terms have to scale with viscosity, density and velocity, which at first sight requires the reassembly at every timestep or even every nonlinear iteration.

- The gradient jump terms couple degrees of freedom which are not coupled in standard Galerkin methods (see Figure 3.1 for an example). This larger stencil of the method increases the number of nonzero entries in the matrix. In two dimensions, the number of nonzero entries is multiplied roughly by two, in three dimensions, there are about three times as many nonzero matrix entries. This property deteriorates the performance of most linear solvers. Incomplete LU factorizations will generate more fillin and use more memory, and data structures and methods based on the standard Galerkin stencil like some multigrid methods cannot be applied in a straightforward manner.

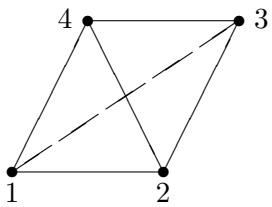


Figure 3.1: Example of additional gradient jump coupling for piecewise linear nodal finite elements on a simple example mesh. The jump of the gradient over edge (2-4) couples degree of freedom 1 with degree of freedom 3, which would be otherwise uncoupled.

In Subsection 3.3.1, we propose a simple method to reduce the assembly costs by updating the matrix less often yet often enough. The issue of additional couplings is adressed in Subsection 3.3.2, where we present a strategy of splitting the stabilization terms into a sum of two terms. One part has the stencil of standard Galerkin discretizations, whereas the other one can be treated explicitly and included in the right hand side vector under certain circumstances.

3.3.1 Reducing Assembly Cost

In an implementation of the interior penalty method, it soon turns out that the assembly of the stabilization terms is the most costly part of the assembly. Because the weight function in the face integrals depends on velocity, density and viscosity, it seems at first sight necessary to reassemble the stabilization matrices whenever one of the three has changed. The weight function however contains also a dimensionless constant γ. Numerical experience as well as analytic considerations show that the method works robustly for a wide range of values for γ. As reusing a stabilization with a slightly different weight amounts to a sligthly different parameter γ which is known to work well, it is enough in practice to reassemble the stabilization matrices only when the weight function has changed considerably.
One therefore needs to detect in a simple and effective way that the weight functions have changed so much that the stabilization terms should be reconstructed. A simple and particularly effective strategy is to rebuild the stabilization at the same time as the preconditioner. In fact the preconditioner is rebuilt when the convergence behaviour with the original preconditioner deteriorates. This indicates usually that the matrix describing the physical characteristics of the system has evolved considerably, such that the old preconditioner is not good enough any more. As it is exactly these physical characteristics defining the weights in the stabilization terms as well, rebuilding the stabilization at the same time as the preconditioner is likely to give good results.
This approach has another advantage: Keeping the stabilization part of the matrix constant for the lifetime of one preconditioner makes the latter better suited for the linear problems.
In Figure 3.2, we compare the total assembly timings for a typical two fluid computation in two dimensions. We consider the average assembly time divided by the dimension M of W_h^1, for the entire matrix $\mathbf{A}$, i.e., containing also the assembly time of the standard Galerkin terms and boundary terms. It can be seen that reusing the stabilization matrix adaptively as described allows to reduce the assembly cost by a factor of about 2 in this case.

3.3.2 Reducing the Stencil

A solution to the problem of the enlarged stencil is proposed and analyzed in [15] for the stationary advection-diffusion-reaction equation. We apply this idea to the pressure stabilization term (2.19):

$$j_p(p,q) = \left\langle \gamma_p \frac{h_f^3}{\max\{h_f\,|\boldsymbol{\beta}|\,,\mu\}}\ [\![\boldsymbol{\nabla} p]\!]_f, [\![\boldsymbol{\nabla} q]\!]_f \right\rangle_{\Gamma_I}$$

We split the interior penalty operator into two parts, one of which gives contributions only to the standard Galerkin stencil. On a given face f let $\boldsymbol{\nabla} p^+|_f$ denote the value $\lim_{\varepsilon\to 0^+} \boldsymbol{\nabla} p(\boldsymbol{x} + \varepsilon \boldsymbol{n}_f)$ and similarly for $\boldsymbol{\nabla} p^-|_f$. The jump of the pressure gradient over the face f defined by (2.9) can be rewritten as $[\![\boldsymbol{\nabla} p]\!]_f = \boldsymbol{\nabla} p^+ - \boldsymbol{\nabla} p^-$. Moreover, we have for $p, q \in V_h^k$

$$[\![\boldsymbol{\nabla} p]\!]_f \cdot [\![\boldsymbol{\nabla} q]\!]_f = [\![\boldsymbol{\nabla} p \cdot \boldsymbol{n}_f]\!]_f\ [\![\boldsymbol{\nabla} q \cdot \boldsymbol{n}_f]\!]_f,$$

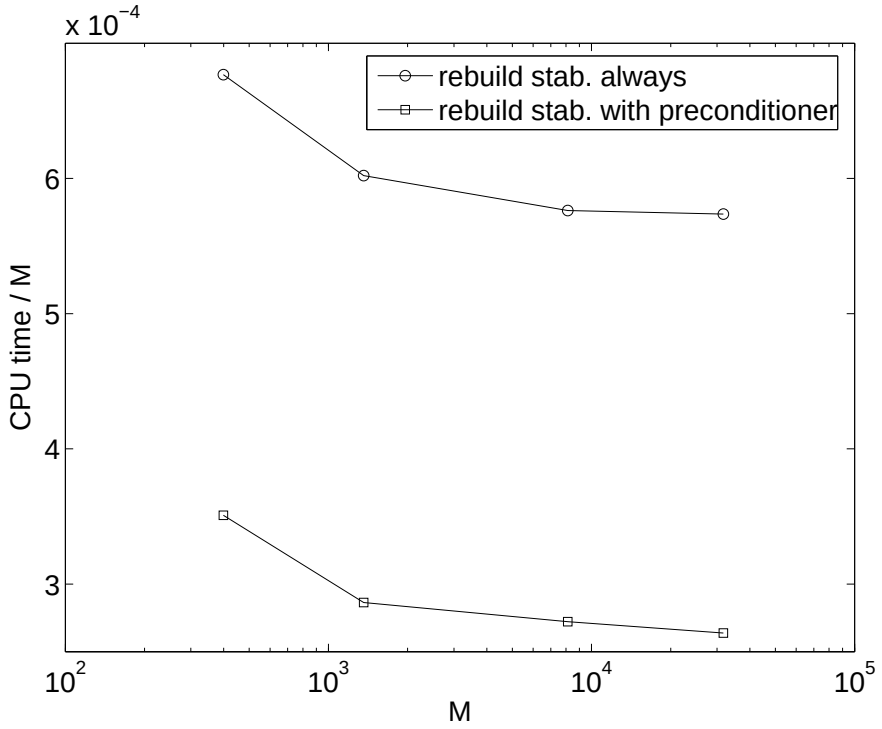


Figure 3.2: Total assembly timings per degree of freedom for a typical two fluid case

because the gradient of a continuous function which is smooth in each element can only jump in the normal direction $\boldsymbol{n}_f$. The splitting reads

$$j_p(p,q) = j_{sG}(p,q) - j_X(p,q),$$

with

$$\begin{aligned} j_{sG}(p,q) &= \langle \tilde{\gamma}_p \boldsymbol{\nabla} p^+ \cdot \boldsymbol{n}_f, \boldsymbol{\nabla} q^+ \cdot \boldsymbol{n}_f \rangle_{\Gamma_I} + \langle \tilde{\gamma}_p \boldsymbol{\nabla} p^- \cdot \boldsymbol{n}_f, \boldsymbol{\nabla} q^- \cdot \boldsymbol{n}_f \rangle_{\Gamma_I}, \\ j_X(p,q) &= \langle \tilde{\gamma}_p \boldsymbol{\nabla} p^+ \cdot \boldsymbol{n}_f, \boldsymbol{\nabla} q^- \cdot \boldsymbol{n}_f \rangle_{\Gamma_I} + \langle \tilde{\gamma}_p \boldsymbol{\nabla} p^- \cdot \boldsymbol{n}_f, \boldsymbol{\nabla} q^+ \cdot \boldsymbol{n}_f \rangle_{\Gamma_I}, \end{aligned}$$

where

$$\tilde{\gamma}_p = \gamma_p \frac{h_f^3}{\max\{h_f \left|\boldsymbol{\beta}\right|, \mu\}}.$$

We note that j_{sG} has the standard Galerkin stencil, whereas j_X has the extended stencil of j_p. We now define a modified bilinear form $\tilde{B}_h$ and a modified linear form $\tilde{f}_h$ as follows:

$$\begin{aligned} \tilde{B}_h[(\boldsymbol{u},p),(\boldsymbol{v},q)] &= a_h(\boldsymbol{u},\boldsymbol{v}) + b_h(p,\boldsymbol{v}) - b_h(q,\boldsymbol{u}) + \theta j_{sG}(p,q), \\ \tilde{f}_h(\boldsymbol{v},q) &= (\boldsymbol{f},\boldsymbol{v}) + \langle \boldsymbol{g}_N, \boldsymbol{v} \rangle_{\Gamma_N} + \langle \sigma\kappa \boldsymbol{n}_\Gamma, \boldsymbol{v} \rangle_\Gamma - \langle \omega \mathbf{P} \boldsymbol{g}_D, \mathbf{P}(2\mu \mathbf{D}(\boldsymbol{v})\boldsymbol{n}) \rangle_{\Gamma_R} \\ &\quad + \langle C_n \boldsymbol{g}_D \cdot \boldsymbol{n}, \boldsymbol{v} \cdot \boldsymbol{n} \rangle_{\Gamma_R} + \langle \omega C_\tau \mathbf{P} \boldsymbol{g}_D, \mathbf{P}\boldsymbol{v} \rangle_{\Gamma_R} \\ &\quad + j_X(\hat{p},q) + (\theta - 1) j_{sG}(\hat{p},q), \end{aligned}$$

where a_h and b_h are defined by equations (2.15) and (2.16), and $\hat{p}$ is some suitable approximation of p. Compared to B_h (2.14), we have omitted the derivative jump penalty terms

of the velocity for simplicity. They are not needed for low Reynolds number computations, and the analogous splitting strategy as for the pressure gradient jump term can be applied to them.
The relaxation parameter θ has to be chosen big enough in order to make the method converge. In [15], convergence of a fixpoint scheme for the scalar advection-reaction-diffusion problem is proven, using the previous iterate as what corresponds to $\hat{p}$ here, provided $\theta > 3$. Also in [15], second order convergence for the time dependent scalar advection-reaction-diffusion problem discretized with BDF2 is proven, using the extrapolation from previous time levels as approximation on the right hand side, which amounts to $\hat{p} = 2p^n - p^{n-1}$ here, for $\theta = 2$.
In analogy to equations (3.3) and (3.4) we can define $\tilde{A}_h$ in function of $\tilde{B}_h$ and $\tilde{F}_h$ in function of $\tilde{f}_h$. The linear functional $\tilde{F}_h$ will depend on some $\hat{p}$. Treating $\hat{p}$ in $\tilde{F}_h$ like $\hat{\boldsymbol{u}}$ is treated in A_h, all iterative schemes presented in Section 3.1 can still be used analogously.
Note that this approach can be combined with the idea from Subsection 3.3.1. This amounts to using for $\tilde{\gamma}_p$ some older value computed with older values of $\boldsymbol{\beta}$ and μ. It is however important to ensure in this case that $\tilde{\gamma}_p$ is the same function in j_{sG} in $\tilde{B}_h$ and in j_{sG} and j_X in $\tilde{f}_h$, otherwise the method will not be consistent. In practice, this means that the function $\tilde{\gamma}_p$ used for the reconstruction of the stabilization matrix has to be stored in some suitable way, in order to be reusable for the evaluation of $\tilde{f}_h$. We choose to approximate the expression for $\tilde{\gamma}_p$ in a finite element space and to store this finite element function, which can then be used for both the matrix assembly and for the right hand side.

Numerical Results

We test this splitting strategy on the Kovasznay benchmark flow problem with $\rho = 1$ and $\mu = 1/40$. It is a two dimensional stationary one fluid flow Navier-Stokes problem with known exact solution. See the description in Subsection 4.1.1 for more details. We use the Oseen part only of the Aitken relaxation scheme from Subsection 3.1.5, because density and viscosity are given constants and we do not need to solve for the transport equation. The tolerance in the Aitken iterations is fixed to $\varepsilon = 0.025h^2$, in order to make sure the error from the nonlinear iteration scheme is at most of the same order as the approximation error from the finite element space. The stationary character is accounted for by taking the limit $\Delta t \to \infty$ in equations (3.3) and (3.4) formally, which cancels several terms. We do not consider gravitation either ($\boldsymbol{g} = \boldsymbol{0}$). We choose linear finite element spaces for both velocity and pressure, and stabilize only the pressure with $\gamma_p = 0.1$. For the initial guess $(\boldsymbol{u}_0, p_0)$, we consider two extreme cases:

1. Choosing $(\boldsymbol{u}_0, p_0)$ as some interpolation of the exact solution will amount to a rather low number of iterations, because in general, one will not know such a good initial guess.

2. On the other hand, choosing $(\boldsymbol{u}_0, p_0) = (\boldsymbol{0}, 0)$ will give a rather high number of iterations, because in a time dependent problem, one will always have a better initial guess, using values from previous timesteps.

In terms of accuracy, the errors for a given mesh are virtually the same for all cases, i.e., for both versions of initial guesses, for all values of θ in the split version of the algorithm as well as for the unsplit version. This can be understood by the fact that the Aitken convergence criterion takes control effectively of this aspect. The results in terms of Aitken iteration counts are presented in Table 3.1 for the first case and in Table 3.2 for the second case.

h	unsplit	$\theta = 1$	$\theta = 1.1$	$\theta = 1.3$	$\theta = 1.5$	$\theta = 1.8$	$\theta = 2$	$\theta = 3$
0.2	6	$\geq$40	11	9	9	9	10	12
0.1	6	14	9	9	10	11	12	14
0.05	6	12	8	9	9	11	11	13
0.02	6	13	9	9	10	11	11	14

Table 3.1: Number of Aitken iterations for different mesh sizes and different values of θ, compared to unsplit method, interpolated solution as initial guess

For the first, we see in Table 3.1 that iteration counts are almost independent of the mesh size. A value of θ just above 1 seems the best choice in order to avoid nonconvergence, but the number of iterations does not increase drastically even for $\theta = 3$. In general, we can say that the number of iterations increases by a factor of 1.5 to 2, compared to the unsplit method.

h	unsplit	$\theta = 1$	$\theta = 1.1$	$\theta = 1.3$	$\theta = 1.5$	$\theta = 1.8$	$\theta = 2$	$\theta = 3$
0.2	8	37	39	$\geq$40	$\geq$40	$\geq$40	$\geq$40	$\geq$40
0.1	9	21	22	26	27	31	34	$\geq$40
0.05	12	$\geq$40	21	23	26	29	32	$\geq$40
0.02	14	$\geq$40	23	24	27	30	32	$\geq$40

Table 3.2: Number of Aitken iterations for different mesh sizes and different values of θ, compared to unsplit method, zero as initial guess

For the second case, we see in Table 3.2 that iteration counts are almost independent of the mesh size for the split method, whereas the unsplit method seems to deteriorate a little with mesh refinement. Again a value of θ just above 1 seems the best choice in order to avoid nonconvergence. In order not to use more than twice the iteration count from the unsplit method, θ should not be chosen bigger than 1.5.
Whether this price is worth to be paid depends on the gain in the data structures and the linear solvers due to the smaller matrix. This difference will be bigger for methods taylored to standard Galerkin stencils than for the standard methods we are applying. Our results indicating the unsplit scheme to be more effective overall are therefore not conclusive in general.

3.4 Incremental Matrix Update in Two Fluid Flow Problems

Consecutive elements in the sequence of matrices described by the iterative schemes in Section 3.1 are closely related. In Subsection 3.2.3, we already exploited the similarity of their spectra for improving the efficiency of solving the associated linear systems by reusing the preconditioner. But also the matrix assembly can be accelerated in this context.

3.4.1 Idea and Algorithm

For simplicity, let us consider the model case of the sequence of scalar mass matrices $\{\mathbf{M}_l\}_l$ associated to some sequence of level set functions $\{\phi_l\}_l$, defined by

$$(\mathbf{M}_l)_{ij} = (\rho(\phi_l)\theta_j, \theta_i),$$

where we recall that $\{\theta_i\}_i$ denotes the standard nodal basis of V_h^k. We note that this is a model case for the first and the third term in the bilinear form a_h (2.15), in the sense that both terms are linear in either ρ or μ, and thus in $H(\phi)$.
The standard way of constructing matrices with variable coefficients is based on a simple recomputation without using previous results. Given the particular situation, we can achieve the goal more efficiently. Formally, we can write

$$\rho_l := \rho(\phi_l) = \rho_- + (\rho_+ - \rho_-)H(\phi_l).$$

Then we have

$$\begin{aligned}\rho_{l+1} &= \rho_- + (\rho_+ - \rho_-)H(\phi_{l+1}) \\ &= \rho_l + (\rho_+ - \rho_-)(H(\phi_{l+1}) - H(\phi_l)) \\ &=: \rho_l + \rho_\delta\end{aligned}$$

The key observation is that ρ_δ has a more compact support than both ρ_l and ρ_{l+1}: It is nonzero only in the area between the old interface defined by ϕ_l and the new one defined by ϕ_{l+1}, i.e., in the area where ϕ_l and ϕ_{l+1} have different signs, whereas the support of both ρ_l and ρ_{l+1} is the entire domain Ω. Provided the displacement of the interface from step l to step $l+1$ is small, the support of ρ_δ is small also. We will show in a complexity analysis below that this assumption is in fact realistic.
The new matrix $\mathbf{M}_{l+1}$ can thus be computed as follows:

$$\begin{aligned}(\mathbf{M}_{l+1})_{ij} &= (\rho_{l+1}\theta_j, \theta_i) \\ &= (\rho_l\theta_j, \theta_i) + (\rho_\delta\theta_j, \theta_i) \\ &= (\mathbf{M}_l)_{ij} + (\rho_\delta\theta_j, \theta_i),\end{aligned}$$

using the old matrix $\mathbf{M}_l$. The computation of the remaining integral can be accelerated exploiting the fact that the density increment is zero in a big part of Ω.
We formulate the algorithm

1. Find the set of elements S_{l+1} where the signs of ϕ_l and ϕ_{l+1} are different:

$$S_{l+1} := \{K \in \mathcal{T}_h : \exists \boldsymbol{x} \in K \text{ such that } \phi_l(\boldsymbol{x})\ \phi_{l+1}(\boldsymbol{x}) < 0\}.$$

2. Integrate $(\rho_\delta\theta_j, \theta_i)$ only over these elements and add the contributions to the old mass matrix:

$$(\mathbf{M}_{l+1})_{ij} = (\mathbf{M}_l)_{ij} + \sum_{K \in S_{l+1}} \int_K \rho_\delta\ \theta_j\ \theta_i\ \mathrm{d}\boldsymbol{x}.$$

Two remarks are in order:

- **Generalization**: The idea of using an old matrix for creating a locally changed new one can be generalized to other situations. One important case is local mesh refinement, where it is possible to remove the contribution from an unrefined element first, and to add the contributions from the refined elements afterwards. Using techniques of static condensation, the matrix data structure can stay unchanged and the matrix update can be done very efficiently.

- **Limitation**: Note that in the case of interfaces moving back and forth all the time during the simulation, there might occur problems of cancellation effects. Although no problems were observed in numerical experiments, special attention has to be payed to this problem, especially in the case of large density ratios $\rho_{\max}/\rho_{\min}$. The loss of precision will be $\log_{10}(\rho_{\max}/\rho_{\min})$ digits. For a typical application like water-air, this means a loss of 3 digits, which appears perfectly acceptable if the simulations are carried out with double precision, i.e., about 15 significant digits.

3.4.2 Complexity Analysis

It is well known and easy to verify that the assembly of a finite element matrix on a discrete space V_h^k of dimension M has computational complexity $\mathcal{O}(M)$, provided the basis functions have local support and this fact is exploited in the assembly. The matrix is assembled taking a loop over all elements in the mesh, and integrate for each element only the basis functions having support on that element. The number of elements is $\mathcal{O}(M)$, and the cost per element is $\mathcal{O}(1)$.
We will assume that the CFL number $c = |\boldsymbol{u}|\,\Delta t/h$ is uniformly bounded, i.e., $\mathcal{O}(1)$. This assumption holds because the splitting of interface advection and flow evolution enforces such a relationship in order to be convergent. The assumption also has to be satisfied in order to balance errors of spatial and temporal discretization as soon as the spatial convergence order is 2 or higher.
Two consecutive level set functions can describe either two different time levels or two different nonlinear iterations. It is clear that the first case is more critical as the level set functions describe approximations of two physically different states, whereas in the second case the physical state approximated by the two level set functions is the same. Given the fact that the interface is transported by the fluid velocity $\boldsymbol{u}$, it moves within one timestep by a distance $\mathcal{O}(|\boldsymbol{u}|\,\Delta t)$. It therefore moves across $\mathcal{O}(|\boldsymbol{u}|\,\Delta t/h) = \mathcal{O}(1)$ elements. The interface Γ is a $d-1$ dimensional surface in $\Omega \subset I\!R^d$, and the number of elements in $\mathcal{T}_h$ constituting Ω will be $\mathcal{O}(M)$, while the number of elements cut by Γ is $\mathcal{O}(M^{(d-1)/d}) = \mathcal{O}(M^{1-1/d})$. The set S_{l+1} is thus constituted by $\mathcal{O}(1)$ layers of $\mathcal{O}(M^{1-1/d})$ elements and therefore of cardinality $\mathcal{O}(M^{1-1/d})$. In analogy to the standard matrix assembly, the cost for computing the new matrix in step 2 of the algorithm is thus $\mathcal{O}(M^{1-1/d})$.
Now let us consider step 1 of the algorithm. It is obvious that it can be achieved by considering every element of the mesh and deciding if it is in S_{l+1} or not. In this case, the complexity is $\mathcal{O}(M)$, with a constant that is small with respect to the one for integration. As the convection term remains to be reassembled at every iteration, which will have a complexity of $\mathcal{O}(M)$, this small additional cost does not do any harm. It shall be noted however that this behaviour could be improved to $\mathcal{O}(M^{1-1/d})$ as well by constructing S_{l+1} using S_l, considering only the elements S_l and elements close to them as member candidates for S_{l+1}.
If the Heaviside function is replaced by a regularization, as proposed in [110] and [111], the

support of ρ_δ will grow. Nevertheless, the transition from 0 to 1 in the regularization is usually chosen to happen over a finite length which is $\mathcal{O}(h)$, i.e., in a layer of $\mathcal{O}(1)$ elements. Therefore, the cardinality of S_{l+1} remains $\mathcal{O}(M^{1-1/d})$. Step 1 of the algorithm may become more expensive as well. It will however not increase the algorithmic complexity.

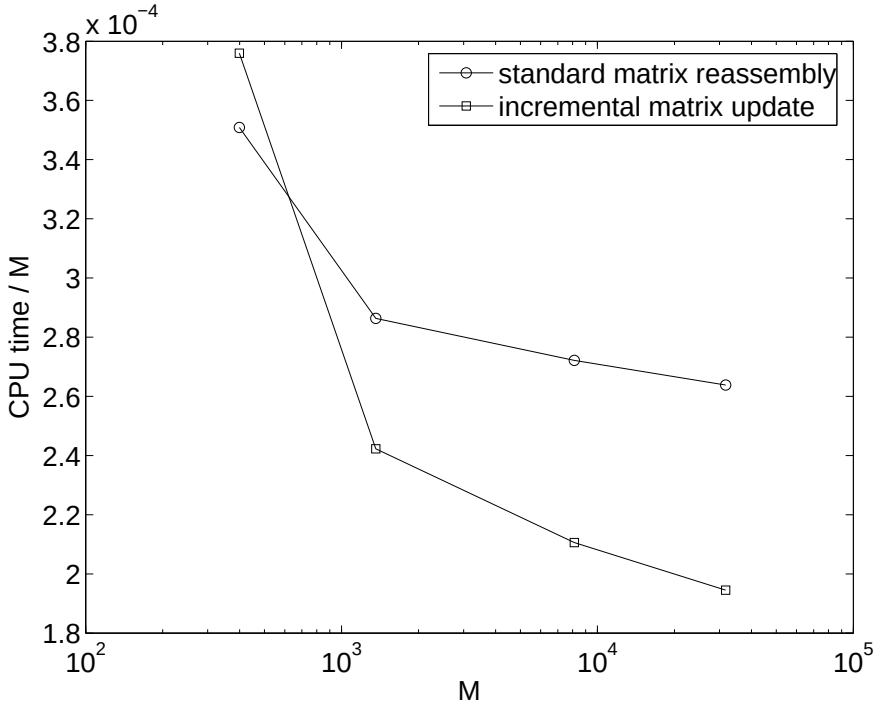


Figure 3.3: Total assembly timings per degree of freedom for a typical two fluid case

Figure 3.3 shows that incremental matrix update allows to reduce the *total* assembly time in a typical two fluid flow problem by more than 20%. This suggests that the dominating cost in assembly is the convective term, which is not specific to two fluid flow. Matrix assembly for two fluid flow can therefore be almost as efficient as for problems with constant density and viscosity.

3.5 Level Set Reinitialization

Many advantages of the level set approach, compared to other front capturing techniques for free surface problems, are based on the smoothness of the transported level set function. We have seen in Section 1.2 that the signed distance function from the interface has particularly advantageous properties in this respect.
Unfortunately, the property $|\boldsymbol{\nabla}\phi| = 1$ of the signed distance function is not preserved under advection of ϕ with the fluid velocity $\boldsymbol{u}$. As a consequence, regions where the level set function becomes too flat appear, and the accuracy in the determination of the interface location deteriorates in these regions. On the other hand, in regions where the level set function becomes too steep, a nearly discontinuous function has to be transported, which calls for application of appropriate discontinuity capturing schemes to reduce numerical oscillations

near the interface.
It was first pointed out in [31] that in order to prevent these effects, a suitable *reinitialization* proceedure should be introduced. Reinitializing the level set function at some given reinitialization time t_r means, given the level set function $\tilde{\phi}(\cdot, t_r)$ which is not a distance function, to find the associated interface $\Gamma(t_r) = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t_r) = 0\}$ first, and to replace $\tilde{\phi}(\cdot, t_r)$ by the signed distance to this interface $\phi(\cdot, t_r) = \text{sdist}(\cdot, \Gamma(t_r))$ afterwards. Different procedures exist to approximate this process efficiently. We will consider them for $\phi \in V_h^1$, i.e., for a piecewise linear approximation.

3.5.1 Desirable Properties of a Reinitialization Procedure

A good reinitialization procedure should satisfy the following criteria:

- **Efficiency**: If the level set function ϕ is approximated in the finite element space V_h^k with dimension N, the reinitialization should have an algorithmic complexity of $\mathcal{O}(N)$ or at most $\mathcal{O}(N \log(N))$.
- **Mass Accuracy**: The interface should not be perturbed too much due to reinitialization. Perturbation of the interface may cause a local mass conservation error. A reasonable criterion is to require this error to be of the same order as the one induced by the approximation of the level set function ϕ in a discrete space. In particular, as ϕ is approximated in V_h^1, the mass error should be of $\mathcal{O}(h^2)$.
- **Distance Accuracy**: The reinitialized level set function should be close to the signed distance function, i.e., $|\boldsymbol{\nabla}\phi| \approx 1$. How close it actually must be depends on the way ϕ is used subsequently, but in general, this part is *not* a crucial requirement. Many standard reinitialization procedures focus too much on distance accuracy and are by consequence too costly or have problems with mass conservation.

We note that mass accuracy is decided only in a small region around the interface, which we denote by Ω_Γ and define as

$$\Omega_\Gamma = \{\boldsymbol{x} \in \Omega : \boldsymbol{x} \in K,\ K \in \mathcal{T}_h,\ K \cap \Gamma \neq \emptyset\},$$

see Figure 3.4. The interface region Ω_Γ is small with respect to $\Omega \in \mathbb{R}^d$, in the sense that $|\Omega_\Gamma| = \mathcal{O}(h\,|\Omega|^{1-1/d})$. It is therefore usual to use some more costly but precise method in the interface region, and another one that is more efficient in the far field region $\Omega_{far} = \Omega \setminus \Omega_\Gamma$.

3.5.2 State of the Art Reinitialization Procedures

Several reinitialization procedures have been proposed in literature. This section gives a short summary of the most relevant ones.

Direct Reinitialization

In the direct approach (see, e.g., [98]), the interface location is computed explicitly. In the case of a piecewise linear approximation $\phi \in V_h^1$, the interface Γ given by $\tilde{\phi}$ can be represented as a mesh of affine $(d-1)$-simplices. For each node $\boldsymbol{x}_j$ of the nodal basis of V_h^1, the signed distance is computed directly by

$$\phi(\boldsymbol{x}_j) = \text{sdist}(\boldsymbol{x}_j, \Gamma),$$

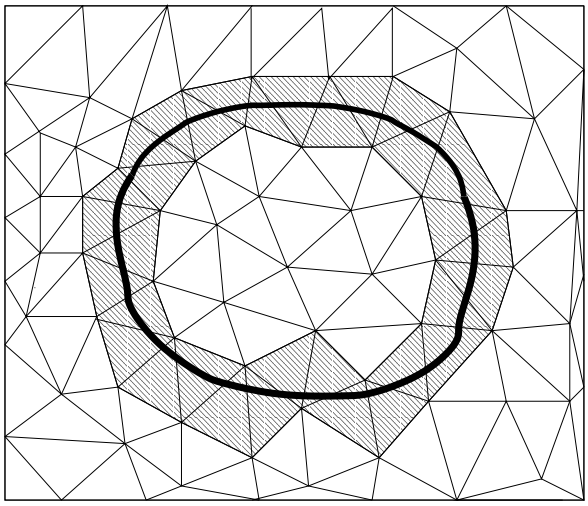


Figure 3.4: Computational domain partitioned in two regions: interface region Ω_Γ (hatched) and far-field region Ω_{far} (white).

i.e., the minimal distance between the node $\boldsymbol{x}_j$ and any node in the mesh representing the interface, multiplied by -1 on Ω^-. Because the full mesh has N nodes and the interface mesh has $\mathcal{O}(N^{1-1/d})$ nodes, overall complexity will be $\mathcal{O}(N^{2-1/d})$, which is excessively expensive. However, the method is very robust and moreover easy to implement. The mass conservation error has optimal order, although it was pointed out in [84] that lower constants can be attained by other methods, specially for smooth interfaces.

Solving a Hamilton-Jacobi Equation

A reinitialization technique based on the solution of an additional partial differential equation was first proposed in [100]. Given $\tilde{\phi}$, the following Hamilton-Jacobi equation has to be solved to steady state in the pseudo time τ:

$$\begin{aligned} \partial_\tau \psi &= \operatorname{sign}(\tilde{\phi})(1 - |\boldsymbol{\nabla}\psi|) \quad \text{in } \Omega \times (0,\infty), \\ \psi|_{\tau=0} &= \tilde{\phi}, \quad \text{in } \Omega, \end{aligned}$$

where sign denotes the signum function. The reinitialized level set function is then given by $\phi = \lim_{\tau\to\infty} \psi$. As shown in [100], a suitable smoothing of the signum function is required for numerical stability reasons. This results in a reduced order of the mass conservation error. Moreover, the additional variable τ typically causes the scheme to have a computational complexity $\mathcal{O}(N^{1+1/d})$.

Fast Marching Method

The fast marching method introduced in [94] is connected to Huygen's principle, which is a construction involving expanding wavefronts, and Dijkstra's method, which is an algorithm for computing smallest cost paths on a network. It works on a discrete set of nodes, which in our case will be the N nodes of a nodal finite element basis. In the finite element context,

neighbor nodes are understood to be nodes in the same element. Because it is necessary to know the value of ϕ on some nodes in order to start the algorithm, it is well suited to be used in the far field region Ω_{far}, using the result of the computation on the interface region Ω_Γ as initial condition.
The fast marching method algorithm is as follows: First, we tag points in the initial conditions as *Accepted*. We then tag as *Close* all neighbor nodes and compute an approximate value of ϕ for them, using only values of *Accepted* neighbor nodes. The exact way of computing this value will be described below. Finally, we tag as *Far* all other grid points. Then the loop is as follows:

1. Let *Trial* be the point in *Close* with the smallest approximated value of $|\phi|$.

2. Add the point *Trial* to *Accepted* and remove it from *Close*.

3. Tag as *Close* all neighbors of *Trial* that are not *Accepted*. If the neighbor is in *Far*, remove it from that list and add it to the set *Close*.

4. Recompute the values of ϕ at all neighbors of *Trial* using only values of *Accepted* neighbor nodes, with the same procedure as in the initialization. Keep the old value of the distance if it was closer to the interface than the newly computed one.

5. If *Close* is not empty, return to step 1.

The efficiency of the algorithm is due to a heapsort technique used to efficiently locate the smallest element in *Close* in step 1. Because exactly one point is accepted in each iteration, the number of loop iterations is bounded by N. Supposing uniform boundedness of the number of neighbors, which is a condition of non-degeneracy of the mesh, it is obvious for all steps of the loop except step 1 that they can be executed in constant time. Storing the set *Close* in a sorted heap data structure, sorting this data structure is of complexity $\log(N)$ and step 1 has therefore the cost $\log(N)$. The overall complexity of the algorithm is thus $\mathcal{O}(N \log(N))$.
The computation of the distance for the neighbors of the nodes in the initial condition and in step 4 using upwind finite differences on a structured grid assures monotonicity of the method. It leads to a quadratic equation to be solved for each value.
The algorithm as presented above generalizes without any problem to unstructured meshes and to any spatial dimension. The crucial point is to define an upwinding distance computation procedure suitable for these cases. The case of unstructured two dimensional meshes has been solved in [64], where a trigonometric description was chosen. In [96], an alternative vectorial description allowed for a generalization to arbitrary dimension. It still leads to a quadratic equation to be solved for each node update. The coefficients of this equation have to be determined from inverting a $d \times d$ matrix. Combining the intuition from [64] with the generality from [96], we suggest in Subsection 3.5.4 a reformulation of the distance computation using Gram-Schmidt orthogonalization, which makes the computation of both matrix inverses and quadratic equation solutions obsolete.
The order of accuracy of the fast marching method is determined by the order of finite differences applied. Because the method is typically applied in the far field where accuracy is not the predominant concern, we will not investigate high order finite differences in what follows.

Interface Local Projection

Interface local projection was introduced in [84] and [21]. It is applicable only to Ω_Γ. Because we approximate all level set functions in V_h^1, the discontinuous function $\bar{\phi} := \tilde{\phi}/|\boldsymbol{\nabla}\tilde{\phi}|$ is a locally exact distance function on each element in the following sense:

- It defines *exactly* the same interface as ϕ, because it is zero at the same values.
- It satisfies $\left|\boldsymbol{\nabla}\bar{\phi}\right| = 1$ *exactly.*

In order to find a continuous approximation $\phi \in V_h^1$, we project $\bar{\phi}$ to V_h^1 with an L^2-projection on Ω_Γ:
Find $\phi_\Gamma \in V_h^1(\Omega_\Gamma)$ such that

$$\int_{\Omega_\Gamma} \phi_\Gamma \psi \, \mathrm{d}\boldsymbol{x} = \int_{\Omega_\Gamma} (\tilde{\phi}/|\boldsymbol{\nabla}\tilde{\phi}|)\psi \, \mathrm{d}\boldsymbol{x}, \quad \forall \psi \in V_h^1(\Omega_\Gamma).$$

This projection requires to solve a mass algebraic problem whose size is proportional to the number of nodes in the interface region Ω_Γ. Its computational cost is therefore negligible in the overall algorithm. Moreover, numerical tests suggest that this projection can be replaced by a less expensive local version without loss of accuracy.
The nodal values of ϕ_Γ have to serve as initial condition for a far field reinitialization procedure like the fast marching method presented before. Denoting by $\tilde{\Gamma}$ the interface defined by $\tilde{\phi}$, the following convergence result is proven in [84]:

$$\|\phi\|_{0,\tilde{\Gamma}} \leq Ch^{3/2}\|\tilde{\phi}\|_{2,\Omega_\Gamma}.$$

This implies $\|\phi\|_{0,\tilde{\Gamma}} \to 0$ when $h \to 0$, and therefore $\tilde{\Gamma} \to \Gamma$.

3.5.3 Applied Methods and Adaptive Scheme

Given the properties of the different methods presented in the previous subsection, we choose to apply the interface local projection method on Ω_Γ, whereas in the far field region Ω_{far} we apply the fast marching method in the reformulation presented in Subsection 3.5.4.
In order to reinitialize the level set function only when the gradient is either too small or too big, we check after every timestep the two following criteria on a set of sample points (e.g. the quadrature points):

$$|\boldsymbol{\nabla}\phi| < 0.2,$$
$$|\boldsymbol{\nabla}\phi| > \phi/h,$$

and reinitialize only if one of the criterions is satisfied somewhere. The first check verifies if the gradient is too small, whereas the second one checks that it is not too big. Note that the second criterion is the most severe near the interface, whereas it will be less critical far from the interface.

3.5.4 Reformulation of the Fast Marching Method in Arbitrary Dimension

We now adress the question of an effective and reliable upwind computation of nodal values of the level set function in the fast marching method. The following derivation allows for

an intuitive understanding, a general formulation in arbitrary dimension d, and an effective implementation of the fast marching method.
Consider the general case where we have to compute an approximate level set value ϕ^* for some given *Close* node $\boldsymbol{x}^*$, which has n *Accepted* neighbor nodes $\boldsymbol{x}_i$, $i = 0 \dots n-1$, with given level set values ϕ_i. We suppose that all these nodes are nodes of the same element K. If this should not be the case, the algorithm has to be applied to all subsets of nodes being in one element, and the value ϕ^* with the smallest magnitude found has to be taken as the final result. We therefore have $n \geq 1$ because $\boldsymbol{x}^*$ is not *Close* if it does not have at least one *Accepted* neighbor, and $n \leq d$ because a simplex element in $\mathbb{R}^d$ has $d+1$ nodes, so one of these nodes has at most d neighbors in this element. See Figure 3.5 for an illustration.

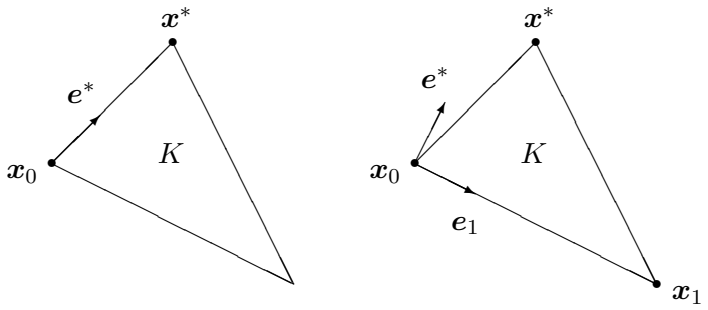


Figure 3.5: Possible situations in two dimensions. Left: one neighbor ($n = 1$), right: two neighbors ($n = 2$).

The (unknown) gradient of the level set function in the element in consideration is a constant vector, because we approximate ϕ with piecewise constants. We denote it by $\boldsymbol{g} := \boldsymbol{\nabla}\phi|_K \in \mathbb{R}^d$. The approximated level set function on K has the form

$$\phi|_K(\boldsymbol{x}) = \boldsymbol{g} \cdot \boldsymbol{x} + c,$$

where c is some constant which can be eliminated using $\phi(\boldsymbol{x}_0) = \phi_0$:

$$\phi|_K(\boldsymbol{x}) = \boldsymbol{g} \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + \phi_0. \tag{3.11}$$

The goal is to determine $\boldsymbol{g}$ such that

$$|\boldsymbol{g}| = 1 \quad \text{and} \tag{3.12}$$
$$\boldsymbol{g} \cdot (\boldsymbol{x}_i - \boldsymbol{x}_0) = \phi_i - \phi_0, \quad i = 1 \dots n-1. \tag{3.13}$$

Once $\boldsymbol{g}$ is known, ϕ^* is obtained from equation (3.11) for $\boldsymbol{x} = \boldsymbol{x}^*$.
The key idea for finding the gradient $\boldsymbol{g}$ is to build an orthogonal coordinate system $\{\boldsymbol{e}_i\}_{i=1}^{n-1}$ centered in $\boldsymbol{x}_0$ of the subspace spanned by the vectors $\boldsymbol{x}_i - \boldsymbol{x}_0$, using Gram-Schmidt orthogonalization of these vectors. By noting $\boldsymbol{x}^* = \boldsymbol{x}_n$ and extending the Gram-Schmidt procedure to $\boldsymbol{x}_n - \boldsymbol{x}_0$, we obtain a vector $\boldsymbol{e}_n = \boldsymbol{e}^*$ which is orthogonal to the subspace spanned by the *Accepted* nodes. In equation (3.13), we express both $\boldsymbol{g}$ and $\boldsymbol{x}_0 - \boldsymbol{x}_i$ using the vectors $\{\boldsymbol{e}_i\}_{i=1}^{n}$. Even for $n < d$, this is possible for the vectors $\boldsymbol{x}_0 - \boldsymbol{x}_i$, because they define the coordinates. Concerning the gradient, it turns out that it is the most conservative assumption to suppose that it can be expressed as a linear combination of the vectors $\boldsymbol{e}_i$ as well. The components

of the gradients in directions $\boldsymbol{e}_i$, $i = 1 \dots n-1$ are determined by equation (3.13). The sum of these components gives a certain magnitude, and if the rest to be added in order to reach magnitude 1 (3.12) is assumed to be the component most collinear to $\boldsymbol{x}^n - \boldsymbol{x}_0$, i.e., in direction $\boldsymbol{e}_n$, it is easy to see from (3.11) that this yields the biggest value of $|\phi^*|$, which is the most conservative assumption. As long as $\boldsymbol{x}^*$ remains in the set *Close*, a smaller value can still be computed using more *Accepted* nodes in a later iteration. See Figure 3.5 for an illustration of the coordinate system $\{\boldsymbol{e}_i\}_{i=1}^n$.
Formulated as an algorithm, the computation of ϕ^* reads as follows:

1. For $i = 1 \dots n$, compute

$$\begin{aligned}
\boldsymbol{v}_i &= \boldsymbol{x}_i - \boldsymbol{x}_0 \\
\boldsymbol{w}_i &= \boldsymbol{v}_i - \sum_{j=1}^{i-1} (\boldsymbol{v}_i \cdot \boldsymbol{e}_j)\boldsymbol{e}_j \\
w_i &= |\boldsymbol{w}_i| \\
\boldsymbol{e}_i &= w_i^{-1} \boldsymbol{w}_i \\
g_i &= \begin{cases} w_i^{-1}(\phi_i - \phi_0 - \sum_{j=1}^{i-1} w_j^{-1}(\phi_j - \phi_0)(\boldsymbol{v}_i \cdot \boldsymbol{e}_j)) & i < n \\ \phi_0 / |\phi_0| \left(1 - \sum_{j=1}^{n-1} g_j^2\right)^{1/2} & i = n \end{cases} .
\end{aligned}$$

2. Compute the gradient as $\boldsymbol{g} = \sum_{i=1}^n g_i \boldsymbol{e}_i$.

3. Compute the level set value as $\phi^* = \phi_0 + (\boldsymbol{x}^* - \boldsymbol{x}_0) \cdot \boldsymbol{g}$.

In order to make sure the upwinding criterion is satisfied, we must check the footpoint $\boldsymbol{x}_f$, i.e., the projection of $\boldsymbol{x}^*$ along the gradient $\boldsymbol{g}$ onto the subspace spanned by the *Accepted* nodes:

$$\boldsymbol{x}_f = \boldsymbol{x}^* - ((\boldsymbol{x}^* - \boldsymbol{x}_0) \cdot \boldsymbol{g})\boldsymbol{g}.$$

We write the footpoint as a linear combination of the *Accepted* nodes:

$$\boldsymbol{x}_f = \sum_{i=1}^{n-1} \lambda_i \boldsymbol{x}_i, \tag{3.14}$$

and solve for the λ_i. Instead of solving the linear system (3.14) with some standard method, we can use again the orthogonal vectors $\boldsymbol{e}_i$ and multiply (3.14) by $\boldsymbol{e}_i$, starting with $\boldsymbol{e}_{n-1}$. We obtain then the values of λ_i one by one. If all $\lambda_i \in [0, 1]$, the linear combination is then a *convex* combination, the footprint lies within the convex hull of *Accepted* points, and the upwinding criterion is satisfied. If this is not the case, the found value of ϕ^* has to be discarded.

3.5.5 Numerical Results

As a test case, we choose a "distorted" circle: On the domain $\Omega = (-1, 1)^2$, the interface is the circle with center $(0, 0)$ and radius $r = 0.5$. As an initial condition, we define $\tilde{\phi} = f(x_1^2 + x_2^2) - f(r^2)$ with $f(\xi) = \xi^{2x_1 + 1/2}$ (see Figure 3.6). Note that the true distance function is obtained with $f(\xi) = \xi^{1/2}$. We use interface local projection in the interface region Ω_Γ and the fast marching method in the far field region Ω_{far}.
In order to evaluate the accuracy, we consider:

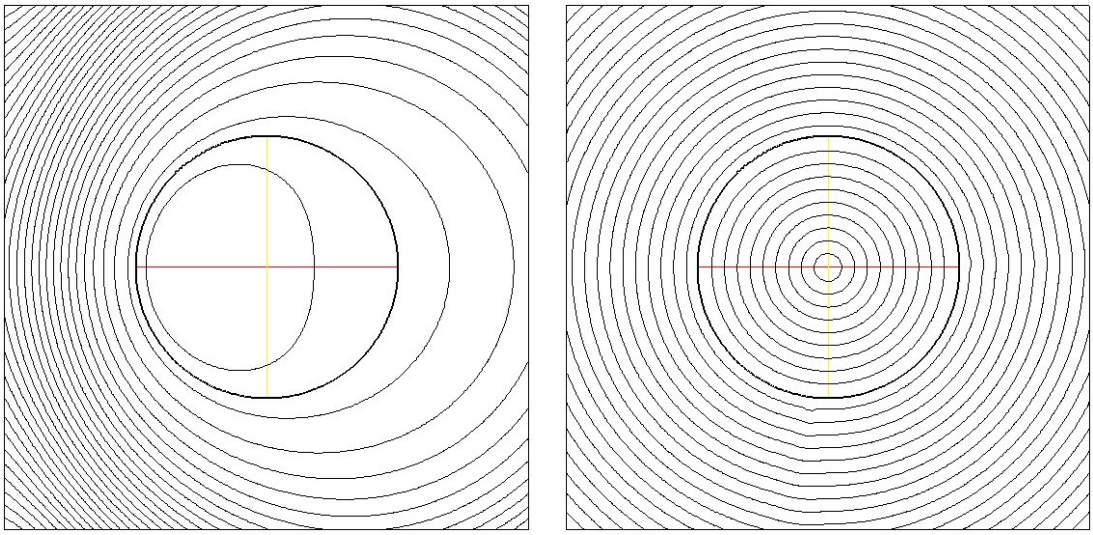


Figure 3.6: Reinitialization test case, left: isolines of $\tilde{\phi}$, right: isolines of ϕ

- the interior of the interface before the reintialization, $\tilde{\Omega}^- = \{\boldsymbol{x} : \tilde{\phi}(\boldsymbol{x}) < 0\}$,
- the interior of the interface after the reintialization, $\Omega^- = \{\boldsymbol{x} : \phi(\boldsymbol{x}) < 0\}$, and
- the region where the reinitialization changes the sign of ϕ, $\Omega_\delta = \{\boldsymbol{x} : \tilde{\phi}(\boldsymbol{x})\ \phi(\boldsymbol{x}) < 0\}$.

Using this notation, we define two error measures:

- the relative mass error: $e_m = (|\Omega^-| - |\tilde{\Omega}^-|)/|\tilde{\Omega}^-|$ and
- the sign change error: $e_s = |\Omega_\delta|$.

The (relative) mass error is a common error measure, but it is insufficient. If $\tilde{\Omega}^-$ is a circle in the lower left corner of the domain and Ω^- a square of the same area in the upper right corner, then the (relative) mass error would be zero. Clearly, we need a better error measure. We therefore consider also the sign change error, i.e., the area of the region where the reinitialization changes the sign of the level set function, which it should not do anywhere.
Figure 3.7 shows that both the relative mass error and the sign change error converge with h^2 and are therefore of optimal order for piecewise linear approximations.
Figure 3.8 shows the total timings of the reinitialization procedure. We can see that the CPU time scales like $N \log(N)$, which is what the complexity analysis lets us expect.
We can therefore conclude that the suggested reinitialization method is optimal with respect to both efficiency and accuracy.

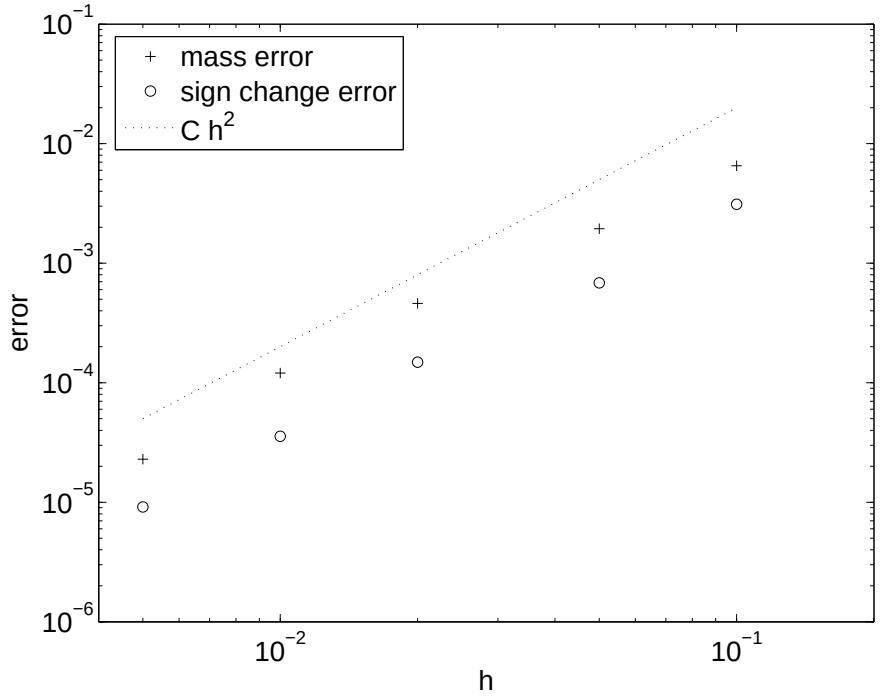


Figure 3.7: Reinitialization errors

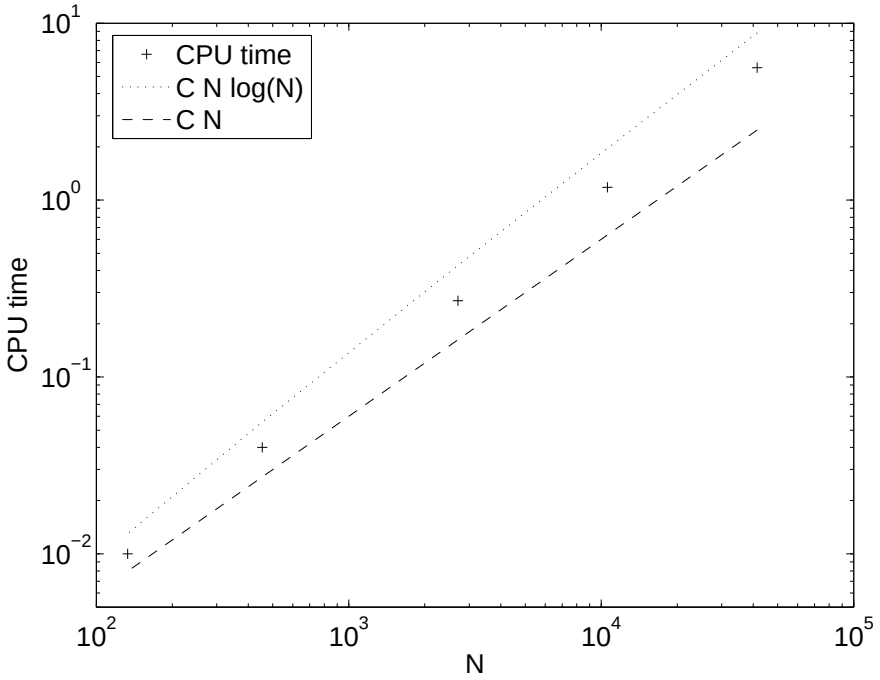


Figure 3.8: Reinitialization timings

Chapter 4

Numerical Results: Accuracy

In this chapter, we consider different problems with nontrivial known exact solutions. The aim is to test the convergence behaviour of the finite element methods introduced in Chapter 2 combined with the iterative schemes from Section 3.1 in simple cases where all kind of errors can be computed because the exact solution is known. More realistic computations are presented in Chapter 5.

4.1 Navier-Stokes with Constant Density and Viscosity

For the Navier-Stokes problem with constant density and viscosity, we compute approximate solutions using the interior penalty formulation from Section 2.2 and compare their convergence behaviour with results using the Taylor-Hood formulation from Section 2.4. We consider a two dimensional stationary problem by Kovasznay [65] in Subsection 4.1.1, a two dimensional time dependent problem introduced by Taylor [105] and popular for benchmarking since Kim and Moin [63], and a three dimensional time dependent problem by Ethier and Steinman [36].

4.1.1 Two Dimensional Stationary Problem

We use the solution by Kovasznay [65] for a first benchmarking of the Navier-Stokes discretizations.

Problem Setting

We consider the following stationary solution to the two dimensional Navier-Stokes equations (1.1),(1.2) with $\rho = 1$ and $\boldsymbol{f} = \mathbf{0}$:

$$\boldsymbol{u}_{ex} = \begin{pmatrix} 1 - e^{\lambda x_1}\cos(2\pi x_2) \\ \frac{\lambda}{2\pi} e^{\lambda x_1}\sin(2\pi x_2) \end{pmatrix},$$
$$p_{ex} = \frac{1}{2}\left(1 - e^{2\lambda x_1}\right),$$

where $\lambda = \frac{1}{2\mu} - \left((2\mu)^{-2} + (2\pi)^2\right)^{1/2}$.
We choose the domain $\Omega = (-0.5, 1) \times (-0.5, 1.5)$ and the viscosity $\mu = 1/40$. Dirichlet boundary conditions with $\boldsymbol{g}_D = \boldsymbol{u}_{ex}$ are imposed on the whole boundary $\partial\Omega$. The stream

lines and the pressure of the exact solution for this viscosity and on the chosen domain are depicted in Figure 4.1.

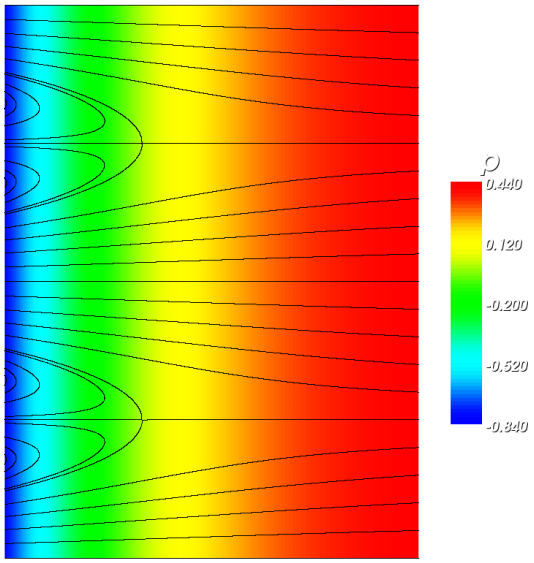


Figure 4.1: Kovasznay flow - stream lines and pressure distribution

Approximation Details

We will look for numerical solutions $(\boldsymbol{u}_h, p_h)$ in 5 different finite element spaces:

- the equal order spaces W_h^1 and W_h^2, using the interior penalty stabilized scheme from Section 2.2 and stabilizing only the pressure with $\gamma_p = 0.1$ for W_h^1 and $\gamma_p = 0.01$ for W_h^2,
- the Taylor-Hood spaces X_h^2 and X_h^3 using the Taylor-Hood scheme from Section 2.4, without any stabilization, and
- the space $\boldsymbol{V}_h^3 \times V_h^1$ for curiosity, using the bilinear forms from the Taylor-Hood scheme, i.e., without any stabilization.

Using a widespread terminology, we will denote a scheme using the space $\boldsymbol{V}_h^k \times V_h^l$ by $\mathbb{P}_k$-$\mathbb{P}_l$ henceforth.
As iterative scheme, we use the Oseen part only of the Aitken formulation from Subsection 3.1.5, because density and viscosity are given constants and we do not need to solve for the transport equation. The tolerance in the Aitken iterations is fixed to $\varepsilon = 0.025h^{k+1}$, in order to make sure the error from the nonlinear iteration scheme is at most of the same order as the approximation error from the finite element space. The stationary character is accounted for by taking the limit $\Delta t \to \infty$ in equations (3.3) and (3.4) formally, which cancels several terms. We do not consider gravitation either ($\boldsymbol{g} = \boldsymbol{0}$ in (3.4)).

Error Quantities

In order to assess the accuracy of the numerical solutions $(\boldsymbol{u}_h, p_h) \in \boldsymbol{V}_h^k \times V_h^l$, we define the following error quantities:

$$e_{\boldsymbol{u},0} = \frac{\|\boldsymbol{u}_h - \boldsymbol{u}_{ex}\|_{0,\Omega}}{\|\boldsymbol{u}_{ex}\|_{0,\Omega}}, \qquad e_{\boldsymbol{u},1} = \frac{\|\boldsymbol{u}_h - \boldsymbol{u}_{ex}\|_{1,\Omega}}{\|\boldsymbol{u}_{ex}\|_{1,\Omega}},$$
$$e_{p,0} = \frac{\|p_h - p_{ex}\|_{0,*,\Omega}}{\|p_{ex}\|_{0,*,\Omega}}, \qquad e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0} = \frac{\|\boldsymbol{\nabla}\cdot\boldsymbol{u}_h\|_{0,\Omega}}{\|\boldsymbol{u}_{ex}\cdot\boldsymbol{n}\|_{0,\partial\Omega}},$$

where $\|\cdot\|_{0,*,\Omega}$ takes into account that the pressure is defined only up to a constant:

$$\|q\|_{0,*,\Omega} := \|q - m_\Omega(q)\|_{0,\Omega},$$
$$m_\Omega(q) := \frac{1}{|\Omega|}\int_\Omega q \, \mathrm{d}\boldsymbol{x}.$$

The function m_Ω denotes the integral mean over the domain Ω.
If the finite element scheme is optimal with respect to interpolation, the error quantities will behave as follows:

$$e_{\boldsymbol{u},0} \leq Ch^{k+1}, \qquad e_{\boldsymbol{u},1} \leq Ch^k,$$
$$e_{p,0} \leq Ch^{l+1}, \qquad e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0} \leq Ch^k.$$

Results and Observations

We compute the four error quantities for four to five different mesh sizes h. The results for the $\mathbb{P}_1$-$\mathbb{P}_1$ elements are traced in Figure 4.2. The slopes of the straight lines fitted to the data give an estimate of the observed convergence order. We observe second order convergence of $e_{\boldsymbol{u},0}$, which corresponds to the estimate from equation (2.21). We also have first order convergence of both $e_{\boldsymbol{u},1}$ and $e_{\boldsymbol{\nabla}\cdot\boldsymbol{u}}$. These three convergence orders are optimal with respect to interpolation. For the pressure, we obtain convergence order 1.8, which is almost optimal (2), and better than the estimate from equation (2.22), which guarantees only order 1.
The results for the $\mathbb{P}_2$-$\mathbb{P}_2$ elements are traced in Figure 4.3. We observe third order convergence of $e_{\boldsymbol{u},0}$ and convergence order two for $e_{\boldsymbol{u},1}$ and $e_{\boldsymbol{\nabla}\cdot\boldsymbol{u}}$, all of which are optimal. The pressure converges almost optimally as well, i.e., with order almost 3.
We can conclude that optimal order convergence of the velocity and almost optimal convergence of the pressure can be expected for interior penalty finite elements in these cases.
The Taylor-Hood elements behave equally well. The results for the $\mathbb{P}_2$-$\mathbb{P}_1$ elements are traced in Figure 4.4. For the classical Taylor-Hood element, all convergence orders are optimal with respect to interpolation, which confirms for the Navier-Stokes case the error estimate (2.23) established for the Stokes problem.
The results for the $\mathbb{P}_3$-$\mathbb{P}_2$ elements are traced in Figure 4.5. Also the higher order Taylor-Hood element exhibits convergence orders that are optimal or even sligthly above for all four error measures.
Finally, we analyze the behaviour of the $\mathbb{P}_3$-$\mathbb{P}_1$ elements which are traced in Figure 4.6. We can see that the convergence order 2 of the pressure is optimal, whereas the convergence order of the velocity error measures is one below the optimality. This can be explained by the fact that the error in the pressure perturbs the velocity equation too much for providing optimal convergence of the velocity.

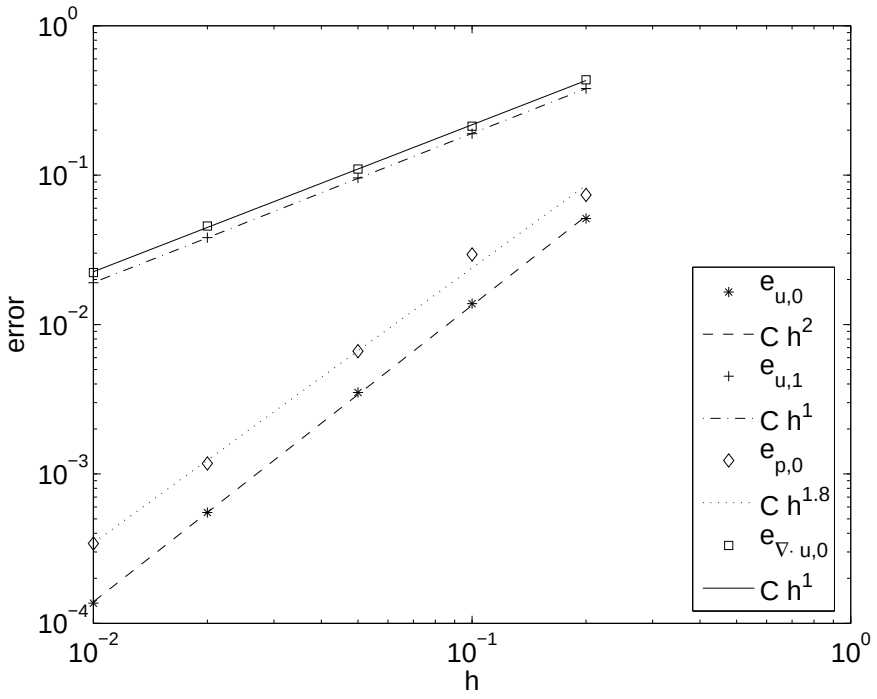


Figure 4.2: Kovasznay flow - $\mathbb{P}_1$-$\mathbb{P}_1$ convergence curves

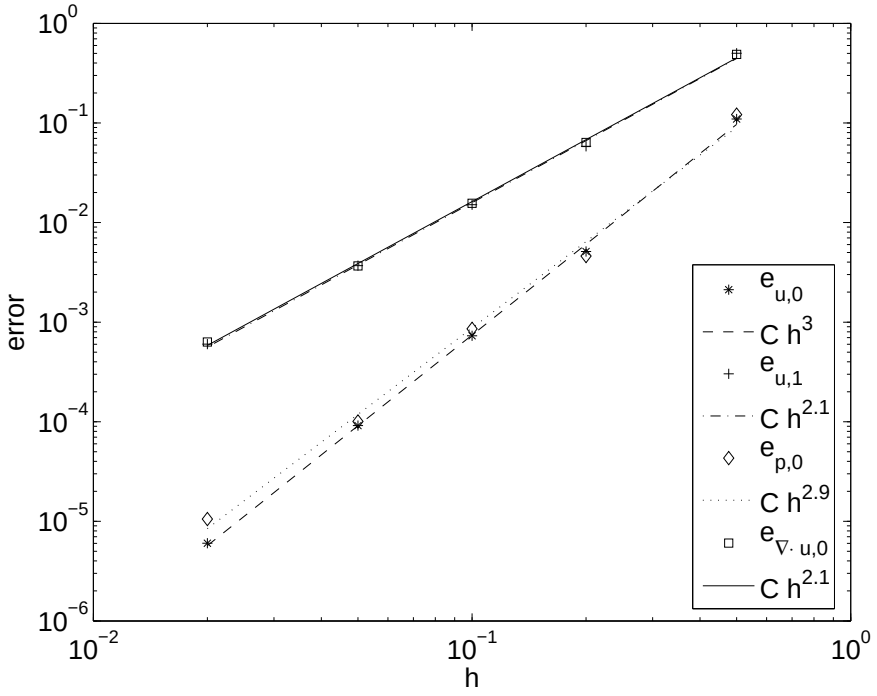


Figure 4.3: Kovasznay flow - $\mathbb{P}_2$-$\mathbb{P}_2$ convergence curves

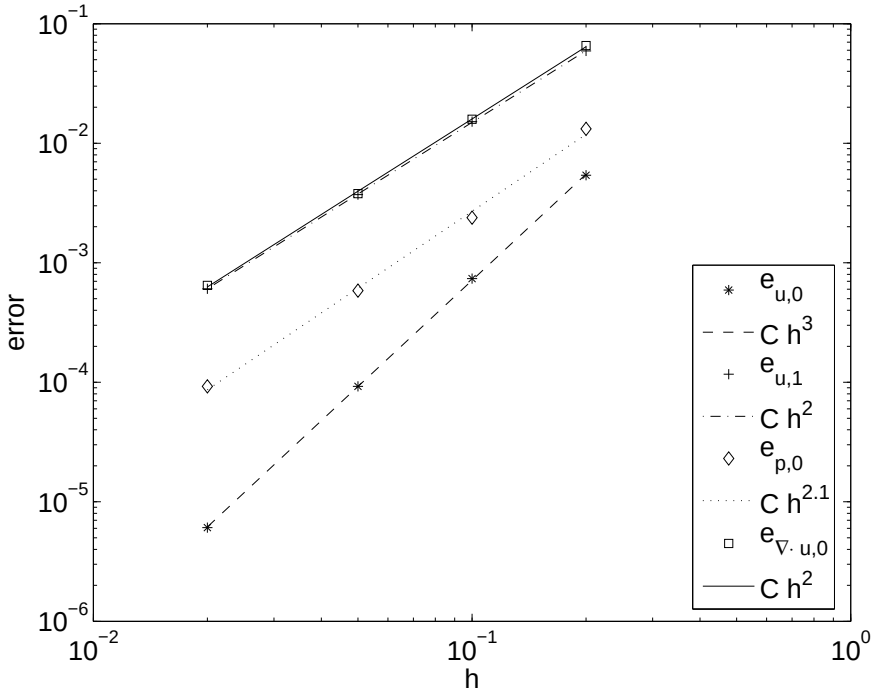


Figure 4.4: Kovasznay flow - $\mathbb{P}_2$-$\mathbb{P}_1$ convergence curves

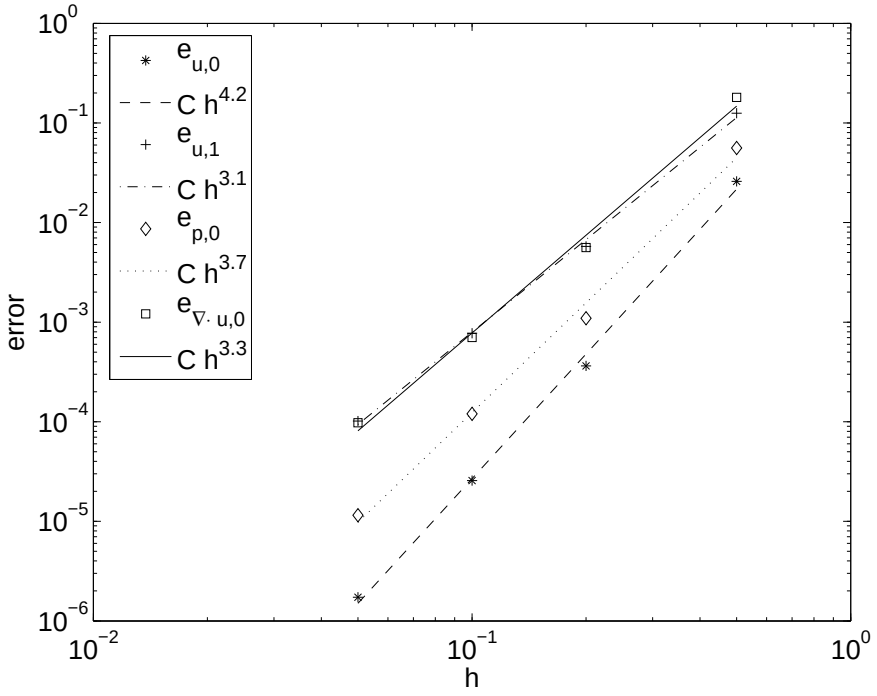


Figure 4.5: Kovasznay flow - $\mathbb{P}_3$-$\mathbb{P}_2$ convergence curves

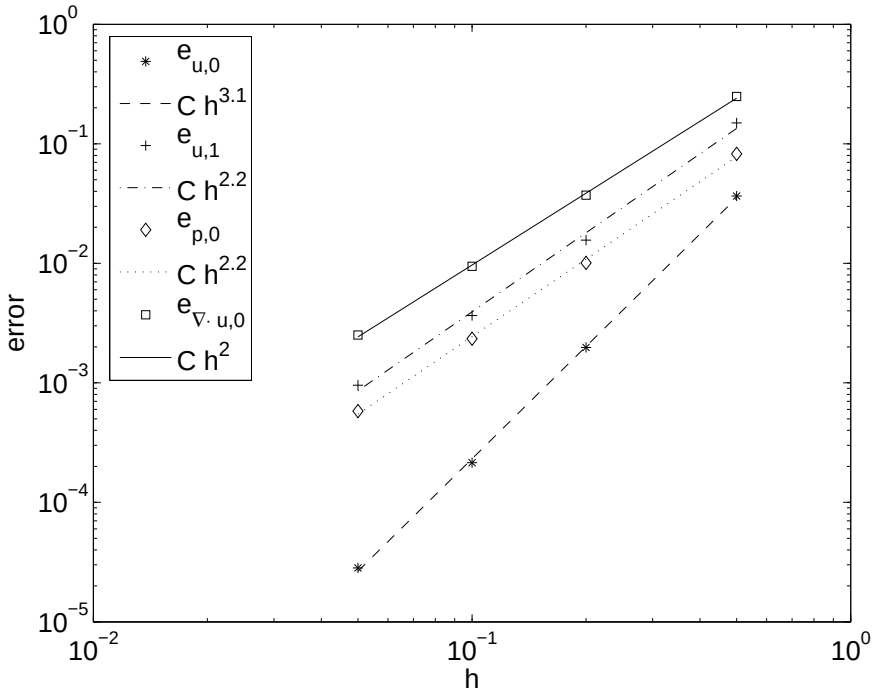


Figure 4.6: Kovasznay flow - $\mathbb{P}_3$-$\mathbb{P}_1$ convergence curves

As a summary, we can see in Table 4.1 that all considered elements beside the non standard $\mathbb{P}_3$-$\mathbb{P}_1$ elements behave optimally in this very simple case. The errors of the $\mathbb{P}_3$-$\mathbb{P}_1$ scheme have actually the same orders for the $\mathbb{P}_2$-$\mathbb{P}_1$ elements, although at a substantially higher computational cost.

discretization	$e_{\boldsymbol{u},0}$	$e_{\boldsymbol{u},1}$	$e_{p,0}$	$e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0}$
$\mathbb{P}_1$-$\mathbb{P}_1$	2.0	1.0	1.8	1.0
$\mathbb{P}_2$-$\mathbb{P}_2$	3.0	2.1	2.9	2.1
$\mathbb{P}_2$-$\mathbb{P}_1$	3.0	2.0	2.1	2.0
$\mathbb{P}_3$-$\mathbb{P}_2$	4.2	3.1	3.7	3.3
$\mathbb{P}_3$-$\mathbb{P}_1$	3.1	2.2	2.2	2.0

Table 4.1: Kovasznay flow - observed convergence orders of different error measures for different discretizations

4.1.2 Two Dimensional Time Dependent Problem

We use the solution by Taylor [105] for benchmarking the Navier-Stokes discretizations combined with the timestepping schemes.

Problem Setting

We consider the following solution to the time dependent Navier-Stokes equations (1.1),(1.2) with $\rho = 1$ and $\boldsymbol{f} = \mathbf{0}$:

$$\boldsymbol{u}_{ex} = \begin{pmatrix} -\cos(\pi x_1)\sin(\pi x_2)e^{-2\pi^2\mu t} \\ \sin(\pi x_1)\cos(\pi x_2)e^{-2\pi^2\mu t} \end{pmatrix},$$
$$p_{ex} = -\frac{\cos(2\pi x_1) + \cos(2\pi x_2)}{4} e^{-4\pi^2\mu t}.$$

We choose the domain $\Omega = (-1, 1)^2$, the time interval $(0, 0.1)$, and the viscosity $\mu = 1$. Dirichlet boundary conditions with $\boldsymbol{g}_D = \boldsymbol{u}_{ex}$ are imposed on the whole boundary $\partial\Omega$. The stream lines and the pressure of the exact solution at time $t = 0$ for this viscosity and on the chosen domain are depicted in Figure 4.7.

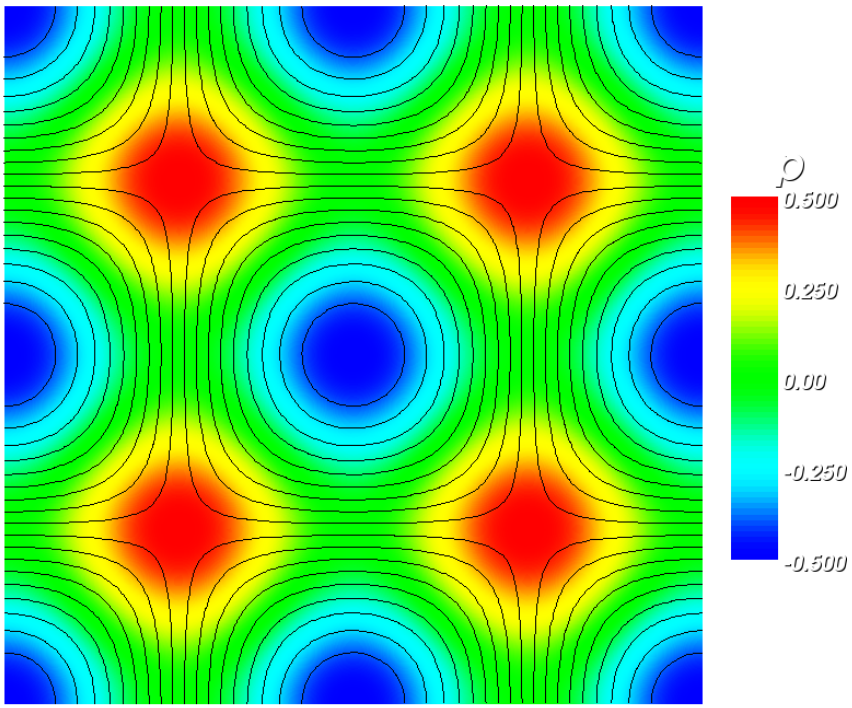


Figure 4.7: Taylor flow - stream lines and pressure distribution at time $t = 0$

Approximation Details

At each timestep t^n, we will look for numerical solutions $(\boldsymbol{u}_h(t^n), p_h(t^n))$ in 3 different finite element spaces:

- the equal order spaces W_h^1 and W_h^2, using the interior penalty stabilized scheme from Section 2.2 and stabilizing only the pressure with $\gamma_p = 0.1$ for W_h^1 and $\gamma_p = 0.01$ for W_h^2,
- the Taylor-Hood space X_h^2 using the Taylor-Hood scheme from Section 2.4, without any stabilization.

As iterative scheme, we use the Oseen part only of the fixpoint formulation from Subsection 3.1.4, because density and viscosity are given constants and we do not need to solve for the transport equation. The tolerance in the fixpoint iterations is fixed to $\varepsilon = 0.025h^{k+1}$, where k is the polynomial order of the velocity approximation. This is to make sure the error from the nonlinear iteration scheme is at most of the same order as the approximation error from the finite element space. We do not consider gravitation ($\boldsymbol{g} = \boldsymbol{0}$ in (3.4)).
To make sure that the timestepping error which is of order Δt^2 does not dominate over the spatial discretization error which is of order h^{k+1} in the best case, we choose $\Delta t \leq Ch$ for the $\mathbb{P}_1$-$\mathbb{P}_1$ elements and $\Delta t \leq Ch^{3/2}$ for the $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ elements.

Error Quantities

In order to assess the accuracy of the numerical solutions $(\boldsymbol{u}_h(t^n), p_h(t^n)) \in \boldsymbol{V}_h^k \times V_h^l$, we define the following error quantities:

$$e_{\boldsymbol{u},0} = \|\boldsymbol{u}_h|_{t=T} - \pi_h^k \boldsymbol{u}_{ex}|_{t=T}\|_{0,\Omega}, \qquad e_{\boldsymbol{u},1} = \|\boldsymbol{u}_h|_{t=T} - \pi_h^k \boldsymbol{u}_{ex}|_{t=T}\|_{1,\Omega},$$
$$e_{p,0} = \|p_h|_{t=T} - \pi_h^l p_{ex}|_{t=T}\|_{0,\Omega}, \qquad e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0} = \|\boldsymbol{\nabla}\cdot\boldsymbol{u}_h|_{t=T}\|_{0,\Omega},$$

where π_h^l denotes the nodal interpolant into the finite element space V_h^l and T is the final time $T = 0.1$.
If the finite element scheme is optimal with respect to interpolation, the error quantities will behave as follows:

$$e_{\boldsymbol{u},0} \leq Ch^{k+1}, \qquad e_{\boldsymbol{u},1} \leq Ch^k,$$
$$e_{p,0} \leq Ch^{l+1}, \qquad e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0} \leq Ch^k.$$

Results and Observations

We compute the four error quantities for four to five different mesh sizes h. The results for the $\mathbb{P}_1$-$\mathbb{P}_1$ elements are traced in Figure 4.8. The slopes of the straight lines fitted to the data give an estimate of the observed convergence order. We observe second order convergence of $e_{\boldsymbol{u},0}$ and first order convergence of $e_{p,0}$ and $e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0}$, which is optimal with respect to interpolation. For the pressure, we obtain convergence order 1.7, which is almost optimal (2), and better than the estimate from equation (2.22), which guarantees only order 1.
The results for the $\mathbb{P}_2$-$\mathbb{P}_2$ elements are traced in Figure 4.9. We observe third order convergence of $e_{\boldsymbol{u},0}$ and second order convergence of $e_{\boldsymbol{u},1}$ and $e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0}$, which is optimal with respect to interpolation. For the pressure, we obtain convergence order 2.2, which is not optimal (3), but still much better than the estimate from equation (2.22), which guarantees only order 1.
The results for the $\mathbb{P}_2$-$\mathbb{P}_1$ Taylor-Hood elements are traced in Figure 4.10. We observe third order convergence of $e_{\boldsymbol{u},0}$ and second order convergence of $e_{\boldsymbol{u},1}$, $e_{p,0}$ and $e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0}$, which are all optimal with respect to interpolation.

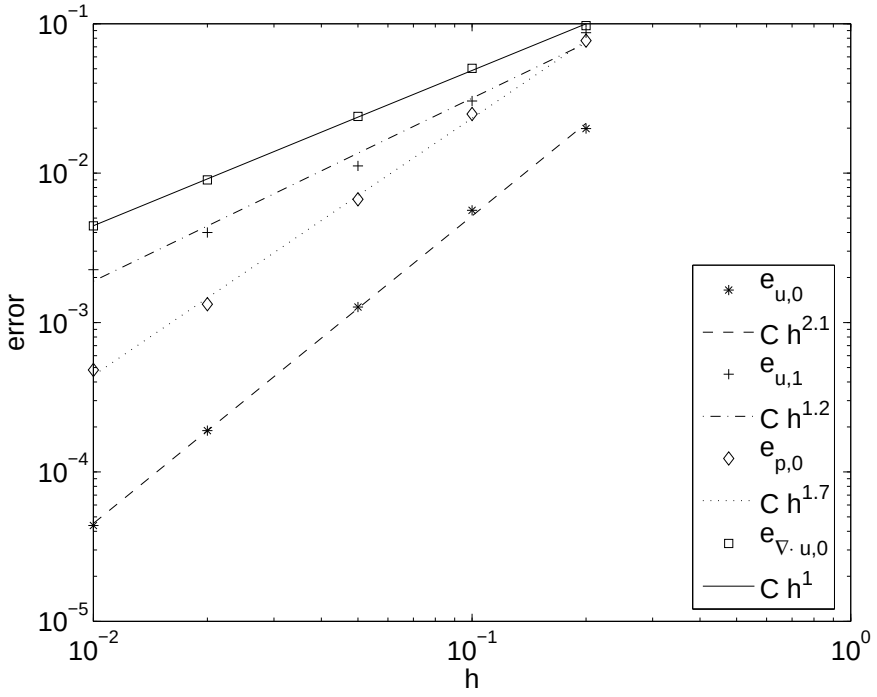


Figure 4.8: Taylor flow - $\mathbb{P}_1$-$\mathbb{P}_1$ convergence curves at time $t = T$

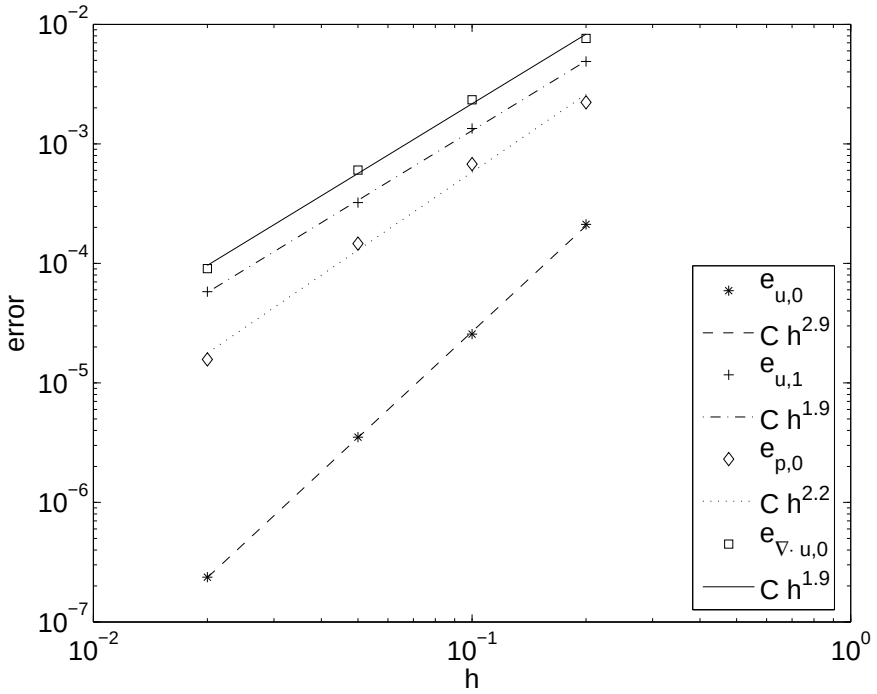


Figure 4.9: Taylor flow - $\mathbb{P}_2$-$\mathbb{P}_2$ convergence curves at time $t = T$

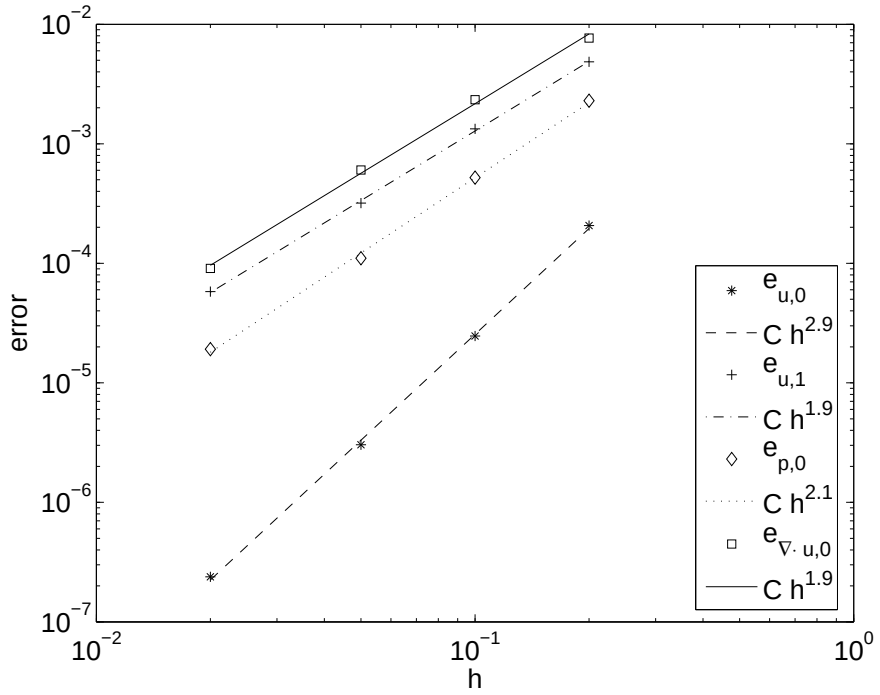


Figure 4.10: Taylor flow - $\mathbb{P}_2$-$\mathbb{P}_1$ convergence curves at time $t = T$

As a summary, we can see in Table 4.2 that the convergence orders of the velocities are optimal with respect to interpolation for the stabilized finite elements, whereas the pressure convergence is sligthly suboptimal. The Taylor-Hood elements in turn converge with optimal orders for all error measures.

discretization	$e_{\boldsymbol{u},0}$	$e_{\boldsymbol{u},1}$	$e_{p,0}$	$e_{\boldsymbol{\nabla}\cdot\boldsymbol{u},0}$
$\mathbb{P}_1$-$\mathbb{P}_1$	2.1	1.2	1.7	1.0
$\mathbb{P}_2$-$\mathbb{P}_2$	2.9	1.9	2.2	1.9
$\mathbb{P}_2$-$\mathbb{P}_1$	2.9	1.9	2.1	1.9

Table 4.2: Taylor flow - observed convergence orders of different error measures for different discretizations

4.1.3 Three Dimensional Time Dependent Problem

We use the solution by Ethier and Steinman [36] for benchmarking the Navier-Stokes discretizations in three dimensions, combined with the timestepping schemes.

Problem Setting

We consider the following parametric smooth solution to the time dependent three dimensional Navier-Stokes equations (1.1),(1.2) with $\rho = 1$ and $\boldsymbol{f} = \mathbf{0}$:

$$\begin{aligned} p_{ex} = \frac{-a^2 e^{-2\mu d^2 t}}{2} \big(& e^{2ax_1} + e^{2ax_2} + e^{2ax_3} \\ & + 2\sin(ax_1 + dx_2)\cos(ax_3 + dx_1) \\ & + 2\sin(ax_2 + dx_3)\cos(ax_1 + dx_2) \\ & + 2\sin(ax_3 + dx_1)\cos(ax_2 + dx_3) \big), \end{aligned}$$

$$\boldsymbol{u}_{ex} = -a\, e^{-\mu d^2 t} \begin{pmatrix} e^{ax_1}\sin(ax_2 + dx_3) + e^{ax_3}\cos(ax_1 + dx_2) \\ e^{ax_2}\sin(ax_3 + dx_1) + e^{ax_1}\cos(ax_2 + dx_3) \\ e^{ax_3}\sin(ax_1 + dx_2) + e^{ax_2}\cos(ax_3 + dx_1) \end{pmatrix},$$

with parameters $a = \pi/4$ and $d = \pi/2$. We choose the domain $\Omega = [-1, 1]^3$ and the time interval $(0, 0.1)$. Dirichlet boundary conditions with $\boldsymbol{g}_D = \boldsymbol{u}_{ex}$ are imposed on the whole boundary $\partial\Omega$. We consider two values of the viscosity, $\mu = 1$ and $\mu = 10^{-5}$. The velocity field of the exact solution at time $t = 0$ (thus independent of μ) on $\partial\Omega$, the boundary of the chosen domain, is depicted in Figure 4.11.

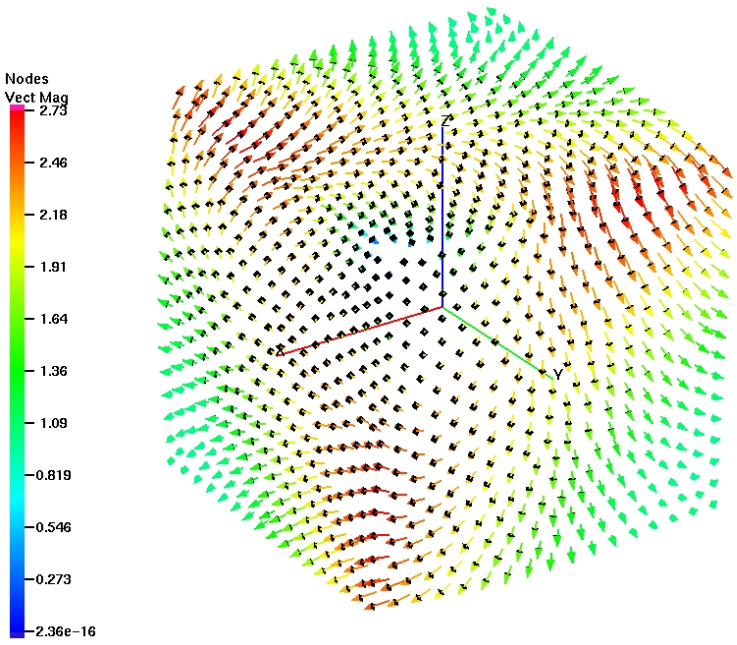


Figure 4.11: Ethier-Steinman flow - velocity field at time $t = 0$

Approximation Details

At each timestep t^n, we will look for numerical solutions $(\boldsymbol{u}_h(t^n), p_h(t^n))$ in the finite element space $\boldsymbol{V}_h^1 \times V_h^1$, using the interior penalty stabilized scheme from Section 2.2, stabilizing the

pressure with $\gamma_p = 0.2$, and imposing the boundary conditions strongly this time. Because we consider also the high Reynolds number case $\mu = 10^{-5}$, we also stabilize the velocity with $\gamma_{\boldsymbol{\beta}} = 0.02$ and $\gamma_{div} = 0.2$. These three stabilization parameters have been found in a parameter study optimizing the convergence behaviour for both cases $\mu = 1$ and $\mu = 10^{-5}$.
As an iterative scheme, we use the Oseen part only of the simple splitting from Subsection 3.1.3, because density and viscosity are given constants and we do not need to solve for the transport equation. We do not consider gravitation ($\boldsymbol{g} = \boldsymbol{0}$ in (3.4)), and choose a fixed timestep of $\Delta t = 0.025$.

Error Quantities

In order to assess the accuracy of the numerical solutions $(\boldsymbol{u}_h, p_h)$, we define the following error quantities:

$$e_{\boldsymbol{u},0} = \frac{\|\boldsymbol{u}_h|_{t=T} - \boldsymbol{u}_{ex}|_{t=T}\|_{0,\Omega}}{\|\boldsymbol{u}_{ex}|_{t=T}\|_{0,\Omega}}, \qquad e_{p,0} = \frac{\|p_h|_{t=T} - p_{ex}|_{t=T}\|_{0,\Omega}}{\|p_{ex}|_{t=T}\|_{0,\Omega}},$$

where T is the final time $T = 0.1$.
If the finite element scheme is optimal with respect to interpolation, the error quantities behave as follows:

$$e_{\boldsymbol{u},0} \leq Ch^2, \qquad e_{p,0} \leq Ch^2.$$

Results and Observations

We compute the two error quantities for four different mesh sizes h and for the two viscosities $\mu = 1$ and $\mu = 10^{-5}$. The results are traced in Figure 4.12. It shows that we have second order convergence of both velocity and pressure in $L^2(\Omega)$ for low and high Reynolds numbers. These convergence orders are optimal with respect to interpolation. Note that also in the high Reynolds number case, the exact solution is laminar. The set of parameters $\gamma_p = 0.2$, $\gamma_{\boldsymbol{\beta}} = 0.02$ and $\gamma_{div} = 0.2$ is therefore suggested as a starting point in the quest of robust stabilization parameters in three dimensional Navier-Stokes flow simulations with continuous interior penalty finite elements of first order.

4.2 Level Set Advection

In order to test the discretization of the level set advection problem separately, we solve a benchmark problem closely related to those proposed in [38]. The problem describes passive transport of the interface by a prescribed divergence free flow field $\boldsymbol{\beta}$.
Let $\Omega = (-1, 1)^2$ be the domain and $(0, T)$ the time interval on which the problem is posed:

$$\begin{aligned} \partial_t \phi + \boldsymbol{\beta} \cdot \boldsymbol{\nabla} \phi &= 0 \quad \text{in } \Omega \times (0, T) \\ \phi &= \phi_0 \quad \text{in } \Omega \text{ at } t = 0 \end{aligned} \tag{4.1}$$

with

$$\boldsymbol{\beta} = \pi \begin{pmatrix} -2x_2(1 - x_1^2) \\ 2x_1(1 - x_2^2) \end{pmatrix} \operatorname{sign}\left(\frac{T}{2} - t\right).$$

As $\boldsymbol{\beta} \cdot \boldsymbol{n} = 0$ on the whole boundary $\partial\Omega$, we have no inflow boundary $\partial\Omega_{in}$ and thus we do not need to specify boundary conditions. We let the velocity field inverse its direction at $t = T/2$,

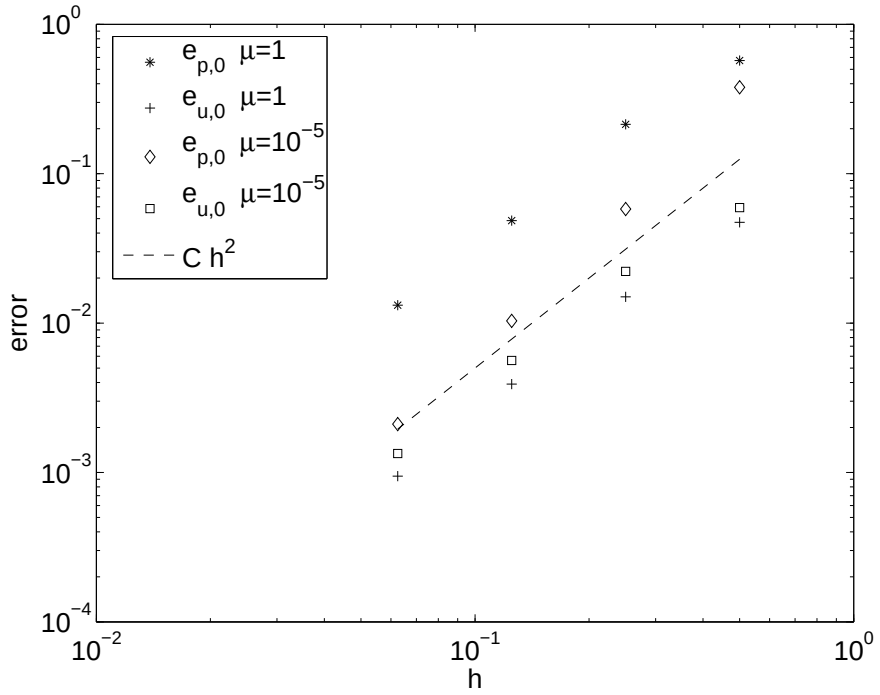


Figure 4.12: Ethier-Steinman flow - $\mathbb{P}_1$-$\mathbb{P}_1$ convergence curves at time $t = T$

such that the motion is inversed and the exact solution satisfies $\phi|_{t=T} = \phi_0$. This allows for easy assessment of the errors accumulated during the simulations.
We discretize equation (4.1) in time by a Crank-Nicolson time discretization scheme [28], and apply the space discretization presented in Section 2.6, approximating $\phi(t^n)$ in V_h^1. The spatial discretization error is of order h^2 for piecewise affine level set functions in the best case, and the time discretization error is of order Δt^2 for the Crank-Nicolson scheme. We therefore choose the timestep such that $\Delta t \leq Ch$ in order to make sure that the timestepping error does not dominate. We will not apply level set reinitialization in order to test the accuracy of the advection scheme alone. Accuracy results for the reinitialization procedure have been presented in Subsection 3.5.5.
We define two error quantities in order to assess the accuracy of the level set advection:

- L^2 error
$$e_{L^2} = \| \phi|_{t=T} - \phi_0 \|_{0,\Omega}$$
- relative mass error
$$e_m = \left| \frac{|\{\boldsymbol{x} : \phi(\boldsymbol{x}, T) < 0\}|}{|\{\boldsymbol{x} : \phi_0(\boldsymbol{x}) < 0\}|} - 1 \right|$$
- sign change error
$$e_{sc} = |\{\boldsymbol{x} : \phi(\boldsymbol{x}, T)\phi_0(\boldsymbol{x}) < 0\}|$$

The initial level set function is given by

$$\phi_0(\boldsymbol{x}) = (x_1^2 + (x_2 - 1/3)^2)^{1/2} - 1/3,$$

defining a circle of radius $1/3$ centered at $(0, 1/3)^\top$. We choose $T = 8$, in order to see enough deformation on the circle. The interface locations at times 0, 1, 2, 3 and 4 are depicted in figure 4.13.

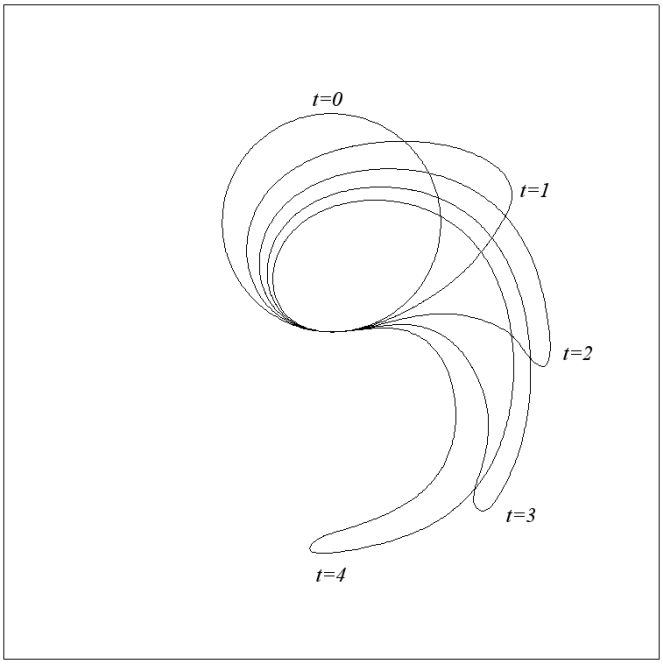


Figure 4.13: Fujima-Ohmori benchmark - interface locations at $t \in \{0, 1, 2, 3, 4\}$

We compute the two error quantities on four different mesh sizes h. The results are traced in Figure 4.14. We observe convergence orders 1.5 and 1.7 for both error quantities, which are in good correspondence with the estimate (2.54) that predicts order $3/2$.

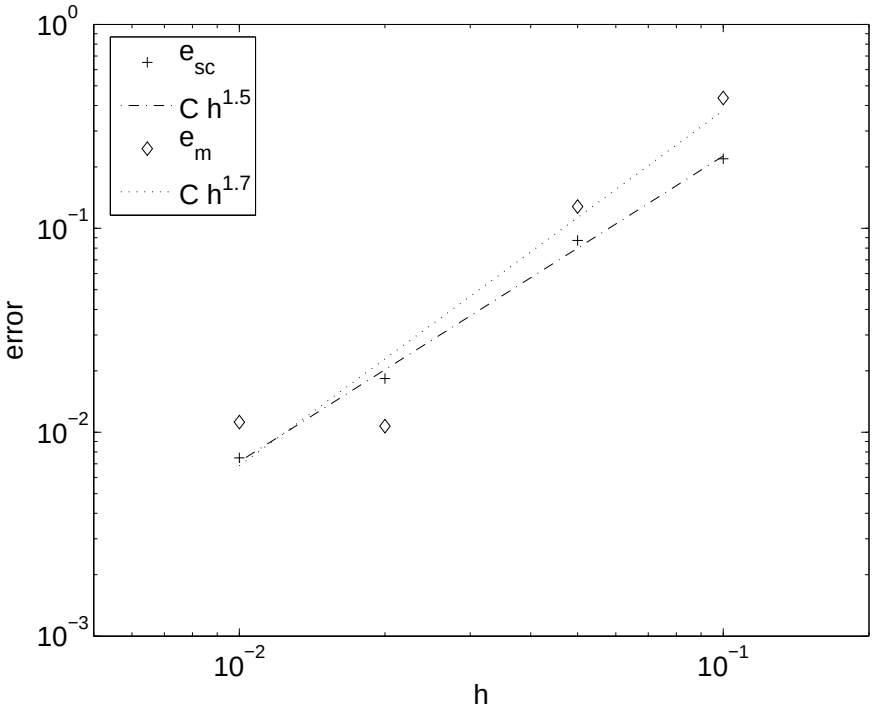


Figure 4.14: Fujima-Ohmori benchmark - convergence curves

Chapter 5

Applications

In this chapter, we consider several benchmark flow problems where no analytic solution is known. The aim is to compare our results to those of well established schemes in the literature. In Section 5.1, we consider a benchmark problem for the one-fluid Navier-Stokes equations in two and three dimensions. In Section 5.2, we solve a benchmark problem for the two-fluid Navier-Stokes equations in two dimensions. Finally, in Section 5.3, we present some computations of free-surface flow with topology changes of the interface, in order to test the ability of the method to cope with this kind of phenomena.

5.1 Laminar Flow around a Cylinder

Schäfer and Turek provide in [93] the definition of a series of test cases involving two and three dimensional laminar flow around a cylinder, along with reference values for some benchmark quantities, established by comparison of results of different research groups.

5.1.1 Definition of the Test Cases

Problem Setting

We consider here the test cases 2D-1 and 3D-Z1 from [93]. The Navier-Stokes equations (1.1), (1.2) are solved on a domain $\Omega \subset I\!R^d$, $d \in 2, 3$. For the test case 2D-1, we have $d = 2$ and the domain Ω is given by Figure 5.1. For the test case 3D-Z1, we have $d = 3$ and the domain

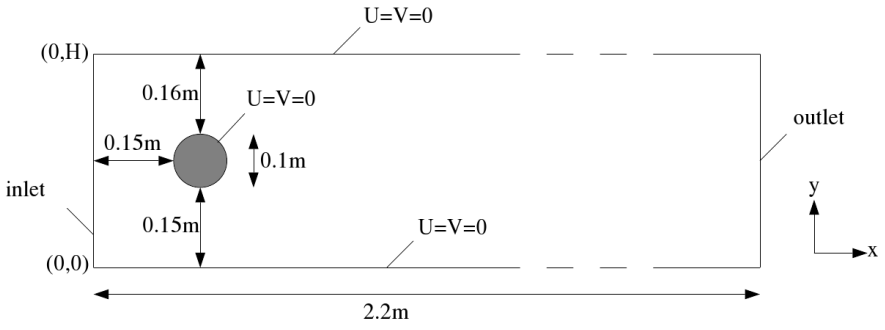


Figure 5.1: 2D cylinder benchmark flow - geometry

Ω is given by Figure 5.2. The density is $\rho = 1.0$ kg/m^3 and the viscosity is $\mu = 10^{-3}$ Pa s,

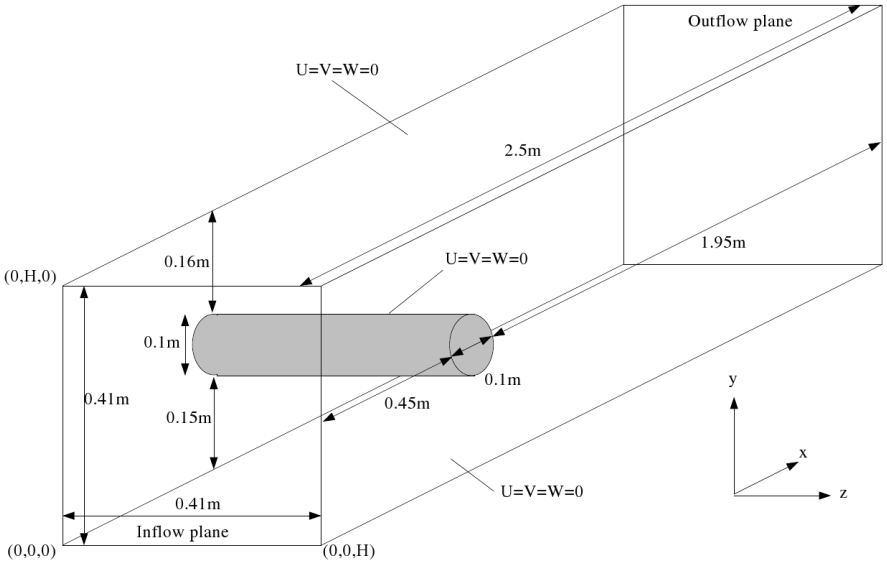


Figure 5.2: 3D cylinder benchmark flow - geometry

both constant throughout in time and space. We denote by H the channel height (and width) $H = 0.41$ m and by D the cylinder diameter $D = 0.1$ m.
On the inflow section, Dirichlet boundary conditions are imposed, with

$$\boldsymbol{g}_D = \left(4U_m x_2(H - x_2)/H^2, 0\right)^\top \quad \text{and} \quad U_m = 0.3 \text{ m/s} \quad \text{for} \quad d = 2,$$
$$\boldsymbol{g}_D = \left(16U_m x_2(H - x_2)x_3(H - x_3)/H^4, 0, 0\right)^\top \quad \text{and} \quad U_m = 0.45 \text{ m/s} \quad \text{for} \quad d = 3,$$

where U_m denotes the maximal inflow velocity. The average inflow velocity is then $\bar{U} = (\frac{2}{3})^{d-1}U_m = 0.2$ m/s in both dimensions, which gives a Reynolds number of $Re = \rho\bar{U}D/\mu = 20$. At this Reynolds number, the flow is stationary. On the outflow section, the benchmark description [93] leaves free choice of boundary conditions. We choose to apply homogeneous Neumann boundary conditions on the outflow section. On the rest of the boundary, homogeneous Dirichlet boundary conditions are imposed.

Benchmark Quantities

The three benchmark quantities are the drag coefficient c_D, the lift coefficient c_L and the pressure difference Δp. For the drag and lift coefficients, we compute the force on surface S of the cylinder, shaded in grey in figures 5.1 and5.2. The cylinder force $\boldsymbol{F}_S$ is given by

$$\boldsymbol{F}_S = \int_S (2\mu \mathbf{D}(\boldsymbol{u}) \cdot \boldsymbol{n} - p\boldsymbol{n}) \, \mathrm{d}S.$$

We note $\boldsymbol{F}_S = (F_D, F_L)^\top$ for $d = 2$ and $\boldsymbol{F}_S = (F_D, F_L, F_z)^\top$ for $d = 3$, where F_D is the drag force and F_L is the lift force. The drag and lift coefficients c_D and c_L are:

$$c_D = \frac{2F_D}{\rho\bar{U}^2 D H^{d-2}} \quad \text{and} \quad c_L = \frac{2F_L}{\rho\bar{U}^2 D H^{d-2}}.$$

The pressure difference is the difference of the pressure between a point $\boldsymbol{x}_a$ before and a point $\boldsymbol{x}_e$ after the cylinder:

$$\Delta p = p(\boldsymbol{x}_a) - p(\boldsymbol{x}_e),$$

where

$$\boldsymbol{x}_a = (0.15, 0.20)^\top \text{ m} \quad \text{and} \quad \boldsymbol{x}_e = (0.25, 0.20)^\top \text{ m} \quad \text{for } d = 2, \text{ and}$$
$$\boldsymbol{x}_a = (0.45, 0.20, 0.205)^\top \text{ m} \quad \text{and} \quad \boldsymbol{x}_e = (0.55, 0.20, 0.205)^\top \text{ m} \quad \text{for } d = 3.$$

5.1.2 Evaluation of the Benchmark Quantities

The integrals for the drag and lift forces are evaluated by numerical integration of the stress on the cylinder surface. This approach has been analyzed in [101]. An alternative approach, first used in [59] and explained in more detail in [8], consists in transforming the surface integrals into integrals over the whole domain. It turns out that the integral forces are simply given by the bilinear form of the finite element method *without boundary terms*, evaluated for $(\boldsymbol{u}, p)$ the solution in consideration and tested with $(\boldsymbol{v}, q)$ such that $\boldsymbol{v}$ is the unit vector in the direction of interest on the part of the boundary of interest S and zero on the rest of the boundary, and $q = 0$. This approach has been extended to the case of stabilized finite element methods in [7].
Note that unlike in two dimensions, the cylinder surface S touches the rest of the boundary $\partial\Omega \setminus S$ in three dimensions. In order to apply the domain integral approach, the test function should then be discontinuous and therefore cannot be in the continuous finite element space. So either the domain integral has to be computed without using the matrix, which makes it way more expensive than surface integration, or the test function has to be approximated in the finite element space, which introduces additional errors. We therefore prefer the straightforward integration of the stress on the cylinder surface.

5.1.3 Two Dimensional Case

Approximation Details

In the two dimensional case, we will look for numerical solutions $(\boldsymbol{u}_h, p_h)$ in 3 different finite element spaces:

- the equal order spaces W_h^1 and W_h^2, using the interior penalty stabilized scheme from Section 2.2 and stabilizing only the pressure with $\gamma_p \in \{0.05, 0.1, 0.2\}$ for W_h^1 and $\gamma_p = 0.01$ for W_h^2;
- the Taylor-Hood space X_h^2 using the Taylor-Hood scheme from Section 2.4, without any stabilization.

Again we denote by $\mathbb{P}_k$-$\mathbb{P}_l$ a scheme using the space $\boldsymbol{V}_h^k \times V_h^l$.
We use the Oseen part only of the Aitken formulation from Subsection 3.1.5 as iterative scheme, because density and viscosity are given constants and we do not need to solve for the transport equation. The tolerance in the Aitken iterations is fixed to $\varepsilon = 0.025h^{k+1}$ in order to make sure the error from the nonlinear iteration scheme is at most of the same order as the approximation error from the finite element space. The stationary character is accounted for by taking the limit $\Delta t \to \infty$ in equations (3.3) and (3.4) formally, which cancels several terms. We do not consider gravitation ($\boldsymbol{g} = \boldsymbol{0}$ in (3.4)).

We carry out simulations on five meshes of different refinement level. These meshes are refined towards the cylinder surface, where more precision is needed, with a mesh width on the cylinder h_{cyl} which is 5 to 10 times smaller than the mesh width h on the rest of the boundary. Some of the mesh characteristics are given in Table 5.1 and the coarsest mesh is depicted in Figure 5.3.

h	h_{cyl}	vertices	edges
0.005	0.0005	69010	205362
0.01	0.001	17523	51739
0.02	0.002	4462	12972
0.03	0.005	1677	4799
0.05	0.01	596	1658

Table 5.1: 2D cylinder benchmark flow - mesh characteristics

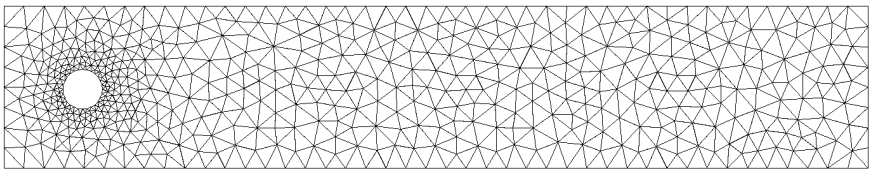


Figure 5.3: 2D cylinder benchmark flow - coarsest mesh

Results and Observations

The general behaviour of the solution is depicted in Figure 5.4. The velocities are highest

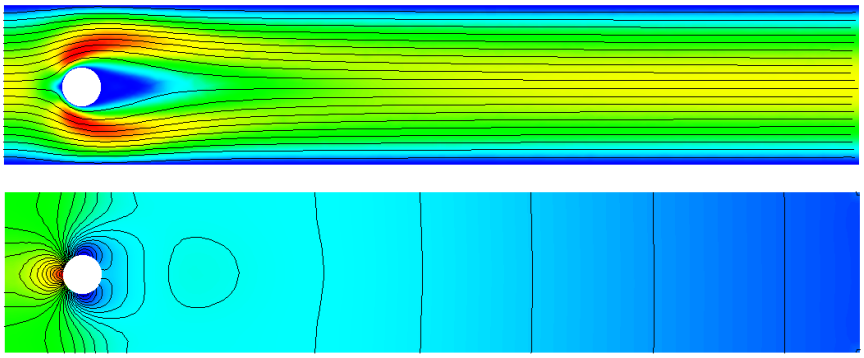


Figure 5.4: 2D cylinder benchmark flow - velocity magnitude and streamlines (top) and pressure distribution and isolines (bottom)

on both sides of the cylinder, whereas a zone of recirculation is formed downstream the cylinder. The pressure decreases from inflow to outflow of course, with strongest gradients around the cylinder. The highest pressure is found at the stagnation point, while the lowest

pressure is not just on the opposite side, but a bit displaced laterally on both sides because of the recirculation. Note that the solution is not perfectly symmetric, because the domain is slightly asymmetric as well.
The values of the benchmark quantities for the three discretizations on several meshes, and for different values of the pressure stabilization parameter for the $\mathbb{P}_1$-$\mathbb{P}_1$ elements, along with bounds for the benchmark quantities from [93], are listed in Table 5.2.

discretization	N	h	c_D	c_L	Δp
lower bound			5.5700	0.0104	0.1172
upper bound			5.5900	0.0110	0.1176
$\mathbb{P}_1$-$\mathbb{P}_1$	207030	0.005	5.5679	0.0095	0.11763
$\gamma_p = 0.2$	52569	0.01	5.5594	0.0077	0.11769
	13386	0.02	5.5421	0.0149	0.11729
	5031	0.03	5.5467	0.0162	0.11640
	1788	0.05	5.6318	0.0238	0.11591
$\mathbb{P}_1$-$\mathbb{P}_1$	207030	0.005	5.5686	0.0094	0.11777
$\gamma_p = 0.1$	52569	0.01	5.5603	0.0075	0.11781
	13386	0.02	5.5411	0.0144	0.11775
	5031	0.03	5.5285	0.0122	0.11802
	1788	0.05	5.4847	0.0198	0.11811
$\mathbb{P}_1$-$\mathbb{P}_1$	52569	0.01	5.5622	0.0074	0.11793
$\gamma_p = 0.05$	13386	0.02	5.5435	0.0143	0.11816
	5031	0.03	5.5314	0.0118	0.11959
	1788	0.05	5.4037	0.0216	0.12040
$\mathbb{P}_2$-$\mathbb{P}_2$	207786	0.01	5.5780	0.0105	0.11753
$\gamma_p = 0.01$	52302	0.02	5.5766	0.0108	0.11765
	19428	0.03	5.5855	0.0094	0.11830
	6762	0.05	5.5604	0.0208	0.11751
$\mathbb{P}_2$-$\mathbb{P}_1$	156047	0.01	5.5775	0.0105	0.11751
	39330	0.02	5.5742	0.0107	0.11745
	14629	0.03	5.5653	0.0077	0.11731
	5104	0.05	5.4974	0.0192	0.11850

Table 5.2: 2D cylinder benchmark flow - results (bounds from [93])

Let us first notice that all discretizations were tested on affine meshes, so the piecewise linear approximation of the cylinder geometry will probably induce dominating errors for the $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ elements.
In order to visualize the convergence behaviour, we apply an affine transformation to the benchmark quantities such that the lower bound is mapped to -1 and the upper bound is mapped to 1. Tracing the absolute values of these normalized errors, we can observe the convergence behaviour more easily. However, we have to consider any absolute value below 1 (dashed lines in plots) as converged.
In Figure 5.5, we trace the normalized errors of the drag coefficient c_D. We can clearly observe that the drag coefficient converges with first order for the $\mathbb{P}_1$-$\mathbb{P}_1$ scheme, independent of the stabilization parameter. For the $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ schemes, we observe at least second

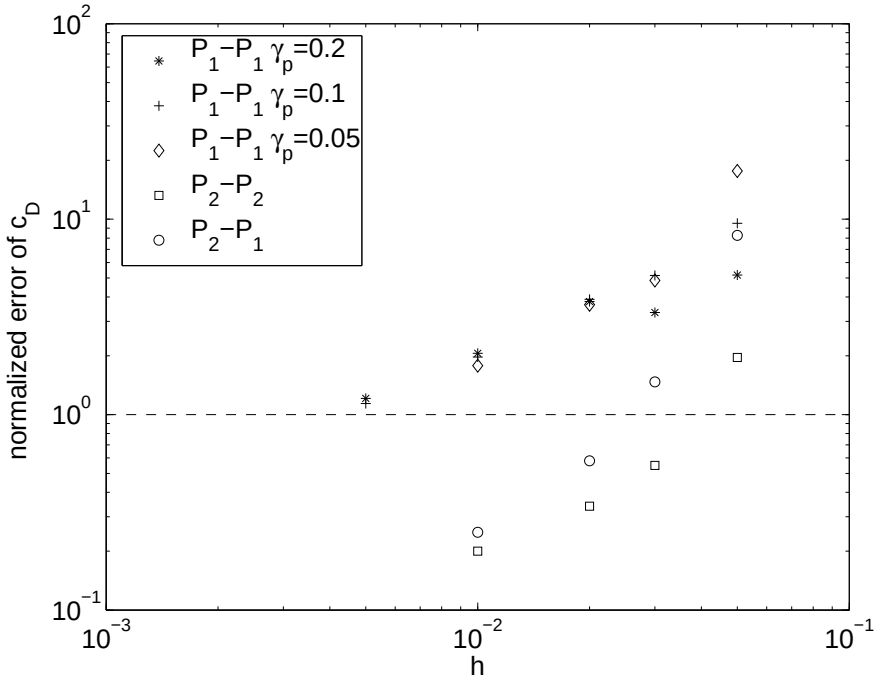


Figure 5.5: 2D cylinder benchmark flow - convergence of c_D

order convergence.
In Figure 5.6, we trace the normalized errors of the lift coefficient c_L. As pointed out also in [93], the lift coefficient is the most difficult to approximate of the three benchmark quantities. Nevertheless, we also can observe first order convergence for the $\mathbb{P}_1$-$\mathbb{P}_1$ scheme. For the $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ schemes, we observe at least second order convergence.
In Figure 5.7, we trace the normalized errors of the pressure difference Δp. The situation for the pressure difference is less clear than for the two other benchmark quantities. The behaviour of the $\mathbb{P}_1$-$\mathbb{P}_1$ scheme is more sensitive to the stabilization parameter, without allowing a clear conclusion about the optimal choice however. The $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ schemes give converged values for quite course meshes already, such that nothing can be said about their convergence order.
As a conclusion, we can say that stabilized first order elements give reliable results for the problem at hand, but the stabilized second order elements as well as the Taylor-Hood elements give have better convergence properties. This is not astonishing as the solution has good regularity properties.

5.1.4 Three Dimensional Case

Approximation Details

One of the conclusions of the comparisons in [93] is that stationary problems are more efficiently solved by stationary solvers than by solving the instationary problem to steady state. The computations in three dimensions are carried out in the finite element library LifeV. This library contains an unsteady Navier-Stokes solver for stable finite element pairs as well as an

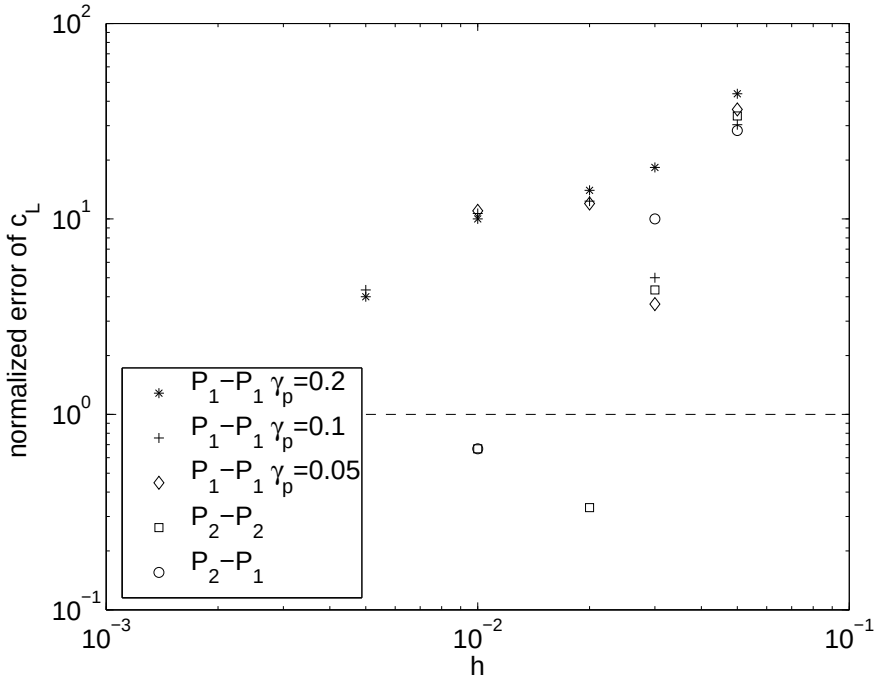


Figure 5.6: 2D cylinder benchmark flow - convergence of c_L

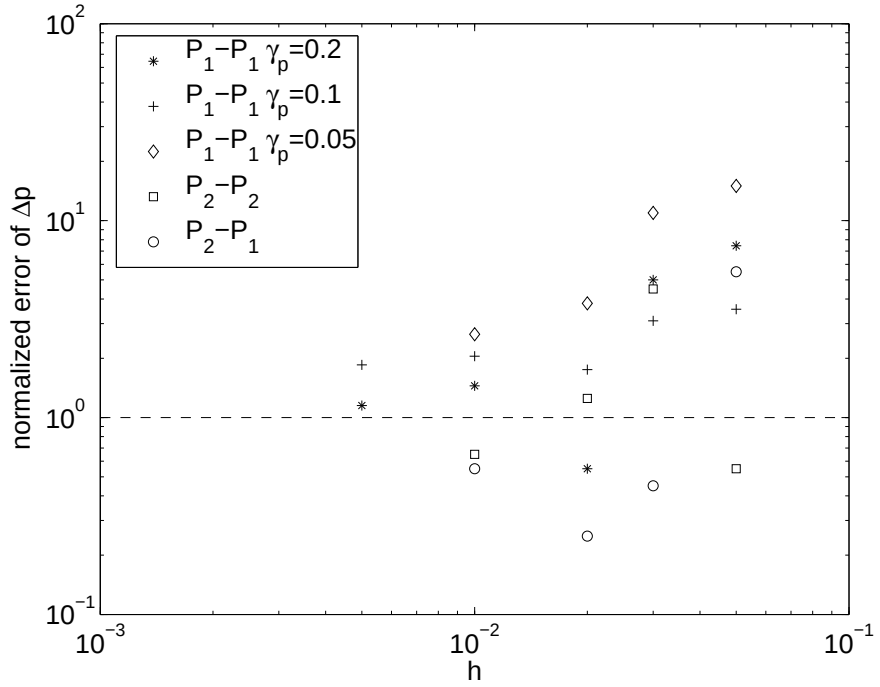


Figure 5.7: 2D cylinder benchmark flow - convergence of Δp

unsteady solver for interior penalty stabilized finite elements with equal order interpolation. In order to test and compare these solvers, we have nevertheless decided to approximate the steady solution by running the time dependent solvers with backward Euler time discretization until $t = T = 10$ s, about where a stationary state is reached.
At each timestep t^n, we will look for numerical solutions $(\boldsymbol{u}_h(t^n), p_h(t^n))$ in 3 different finite element spaces:

- the equal order spaces W_h^1 and W_h^2, using the interior penalty stabilized scheme from Section 2.2 with boundary conditions imposed strongly, stabilizing the pressure with $\gamma_p = 1/32$, the convective term with $\gamma_{\boldsymbol{\beta}} = 1/32$ but not the divergence ($\gamma_{\text{div}} = 0$),
- the Taylor-Hood space X_h^2 using the Taylor-Hood scheme from Section 2.4, with boundary conditions imposed strongly and without any stabilization.

Note that for $Re = 20$, a stabilization of the convective term is not strictly necessary, and one could set $\gamma_{\boldsymbol{\beta}} = 0$ also for the equal order spaces. We choose to set $\gamma_{\boldsymbol{\beta}} = 1/32$ in order to see how much it influences the result in case we want a uniform choice of the stabilization for all Reynolds numbers.
We use the Oseen part only of the simple splitting from Subsection 3.1.3 as iterative scheme, because density and viscosity are given constants and we do not need to solve for the transport equation. We do not consider gravitation ($\boldsymbol{g} = \boldsymbol{0}$ in (3.4)).
The solver for the Taylor-Hood finite element spaces applies an algebraic factorization strategy for solving the linear systems.
The meshes used for the convergence study are refined towards the cylinder, with a mesh width on the cylinder h_{cyl} which is 12.5 times smaller than the mesh width h on the rest of the boundary. Some details on the meshes are listed in Table 5.3.

h	h_{cyl}	vertices	cells	edges
0.025	0.002	168020	816599	
0.05	0.004	38054	180411	233826
0.1	0.008	9111	42930	55784
0.2	0.016	2085	9400	12445

Table 5.3: 3D cylinder benchmark flow - mesh characteristics

The timestep Δt is chosen as

$$\Delta t = h \cdot 1 \text{ s/m} = 12.5\ h_{cyl} \cdot 1 \text{ s/m}.$$

With a maximal inflow velocity of 0.45 m/s, this would in theory give a maximal Courant number of $0.45 \cdot 12.5 = 5.625$. Note that the effective maximal Courant number is lower in practice, as the velocity is zero where the mesh is finest (i.e., on the cylinder).

Results and Observations

The general behaviour of the solution is depicted in Figures 5.8-5.12. Figure 5.8, shows the velocity field at $z = 0.205$. As in the two dimensional case, the velocities are highest on both sides of the cylinder, whereas a zone of recirculation is formed downstream the cylinder.
The velocity field at $z = 0.075$ is shown in Figure 5.9. The behaviour is essentially the same,

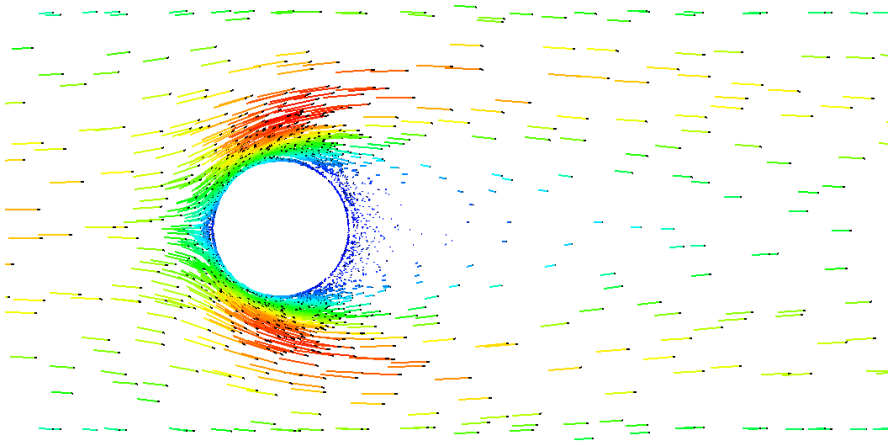


Figure 5.8: 3D cylinder benchmark flow - velocity field at $z = 0.205$

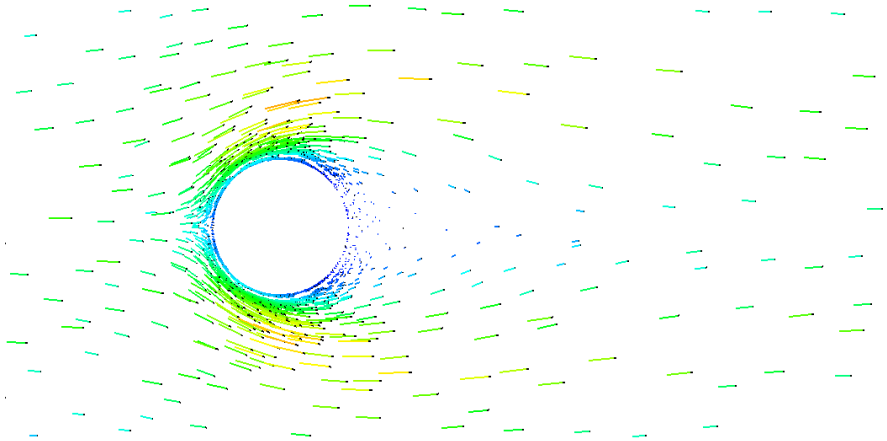


Figure 5.9: 3D cylinder benchmark flow - velocity field at $z = 0.075$

as at $z = 0.205$, but at a lower magnitude.
The velocity field in a section at $y = 0.2$ through the cylinder is shown in Figure 5.10. We can

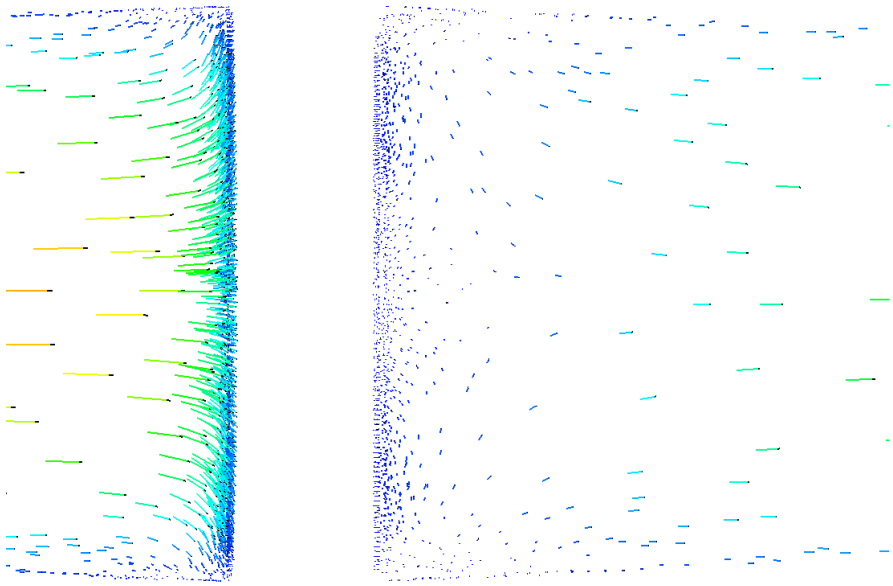


Figure 5.10: 3D cylinder benchmark flow - velocity field at $y = 0.2$

see the parabolic inflow profile upstream the cylinder, and the stagnation effect. Moreover, we see that before the cylinder, the fluid rises in the upper part and it sinks in the lower part. This means that the velocity profile of the flattens while it is flowing around the cylinder.
Figure 5.11 shows the velocity field at $y = 0.1$. Also here, we can see the acceleration of the fluid around the cylinder.
Finally, the pressure at $z = 0.205$ is depicted in Figure 5.12. As in the two dimensional case, we see the highest pressure at the stagnation point, and two pressure minima laterally on the cylinder.
The values of the benchmark quantities for the three discretizations on several meshes, along with bounds from [93] and reference values from [8], are listed in Table 5.4.
Let us first notice that all discretizations were tested on affine meshes, so the piecewise planar approximation of the cylinder geometry will probably induce dominating errors for the $\mathbb{P}_2$-$\mathbb{P}_2$ and the $\mathbb{P}_2$-$\mathbb{P}_1$ elements.
The stabilized $\mathbb{P}_1$-$\mathbb{P}_1$ elements yield values for the drag coefficient and for the pressure difference which are within the bounds or very close for all but the coarsest meshes. The lift coefficient seems well approximated on the finest mesh only, being too large though. As we can see, the presence of the stabilization for the convective term does not have a big influence on the obtained values.
The stabilized $\mathbb{P}_2$-$\mathbb{P}_2$ elements give values within the bounds for drag coefficient and pressure difference already on the coarsest mesh. Given the general difficulty of approximating the lift on coarse meshes (see also [93]), nothing can be said about the lift. Running a finer mesh was not possible due to memory restrictions.

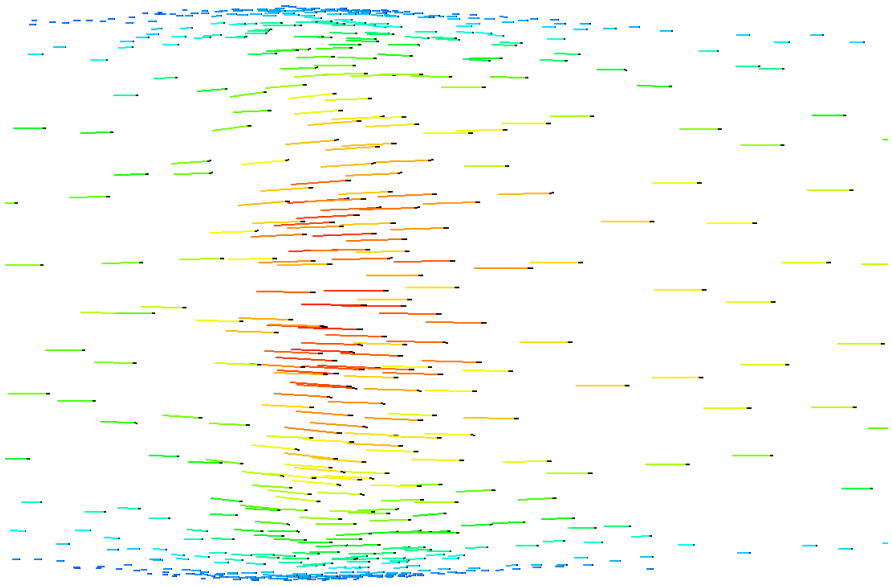


Figure 5.11: 3D cylinder benchmark flow - velocity field at $y = 0.1$

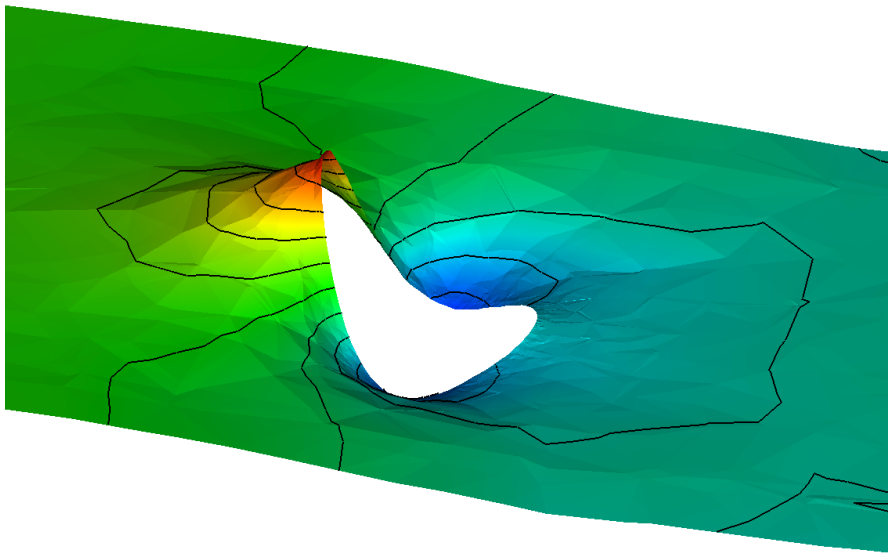


Figure 5.12: 3D cylinder benchmark flow - pressure at $z = 0.205$

discretization	N	$h = \Delta t$	c_D	c_L	Δp
lower bound			6.05	0.0080	0.165
reference			6.185	0.0094	0.1710
upper bound			6.25	0.0100	0.175
$\mathbb{P}_1$-$\mathbb{P}_1$	672080	0.025	6.25	0.0111	0.173
$\gamma_\beta = 1/32$	152216	0.05	6.27	0.0205	0.176
	36444	0.1	6.25	-0.0039	0.171
	8340	0.2	6.54	0.0200	0.177
$\mathbb{P}_1$-$\mathbb{P}_1$	672080	0.025	6.24	0.0116	0.173
$\gamma_\beta = 0$	152216	0.05	6.23	0.0221	0.175
	36444	0.1	6.16	-0.0082	0.171
	8340	0.2	6.30	0.0337	0.176
$\mathbb{P}_2$-$\mathbb{P}_2$	58120	0.2	6.24	0.0174	0.174
$\mathbb{P}_2$-$\mathbb{P}_1$	853694	0.05	6.18	0.0090	0.171
	203796	0.1	6.17	0.0097	0.169
	45675	0.2	6.18	-0.0052	0.169

Table 5.4: 3D cylinder benchmark flow - results (bounds from [93], reference values from [8])

The $\mathbb{P}_2$-$\mathbb{P}_1$ Taylor-Hood elements yield values for the drag coefficient and for the pressure difference which are within the bounds from the coarsest mesh on. The lift coefficient is within bounds if the mesh is not too coarse.
The $\mathbb{P}_2$-$\mathbb{P}_1$ elements clearly approximate best the reference values from [8].

5.2 Rising Bubble with Surface Tension

Although numerical simulation of incompressible two-phase flows is maturing at a rapid rate, not many *quantitative* numerical benchmark problems have been proposed that far. To our knowledge, the only rigorous such proposition can be found in the forthcoming paper [57]. The authors propose as test case a rising bubble with two different sets of physical parameters. They provide a definition of a set of benchmark quantities and detailed numerical results for three codes with different discretizations. In this section, we validate our implementation by reproducing the results from [57]. The simulation runs were carried out in the framework of a semester project [92] supervised by the author of this thesis.

5.2.1 Definition of the Test Cases

This subsection describes the governing equations and defines the test cases and benchmark quantities to be used for the validation of the implementation.

Problem Setting

The flow is governed by the Navier-Stokes equations (1.1)-(1.2) with variable density and viscosity, and the interface evolution is modeled by the level set evolution equation (1.15). Interface conditions (1.9) and (1.10) hold at the bubble surface. We solve these equations on the domain $\Omega = (0,1) \times (0,2) \subset I\!R^2$ and in the time interval $(0,T)$ with $T = 3$. The initial

configuration, see Figure 5.13, consists of a circular bubble of radius $r_0 = 0.25$ centered at $(0.5, 0.5)$. This configuration is described by the signed distance function $\phi_0 = ((x_1 - 1/2)^2 + (x_2 - 1/2)^2)^{1/2} - r_0$. Like in [57] and in Section 2.5, we will note Ω_1 instead of Ω^+ and Ω_2

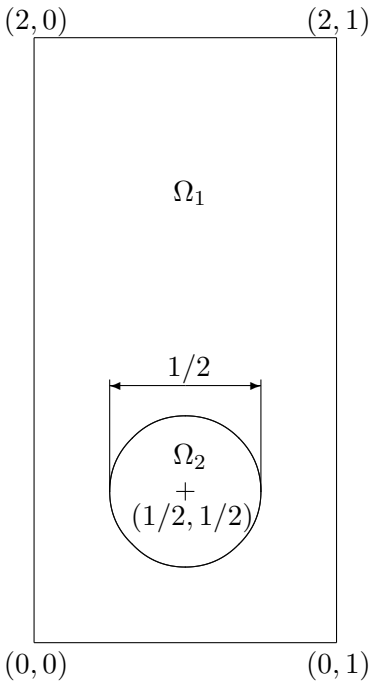


Figure 5.13: Domain and initial configuration for both test cases

instead of Ω^-, and use according indices for density and viscosity. The fluid is at rest initially ($\boldsymbol{u}_0 = \mathbf{0}$).
On the vertical boundaries, we apply free slip boundary conditions, whereas at the horizontal boundaries, no-slip boundary conditions are imposed. In our weak formulation, this means that $\Gamma_R = \partial\Omega$ with $\omega = 1$ on $(0, 1) \times \{0, 2\}$ and $\omega = 0$ on $\{0, 1\} \times (0, 2)$, with $\boldsymbol{g}_D = \mathbf{0}$.
The Reynolds number Re relates inertial effects to viscous effects, and the Eötvös number relates gravitational forces to surface tension effects. We use the definitions from [57]:

$$Re = \frac{\rho_1 \sqrt{|\boldsymbol{g}|}(2r_0)^{3/2}}{\mu_1}, \qquad Eo = \frac{4\rho_1 |\boldsymbol{g}| r_0^2}{\sigma},$$

where $\boldsymbol{g} = (0, -0.98)^\top$ is the acceleration of gravity which will drive the lighter bubble to rise in the heavier surrounding fluid. Together with the density ratio ρ_1/ρ_2 and the viscosity ratio μ_1/μ_2, these dimensionless numbers allow to classify the two test cases. The physical parameters and dimensionless numbers which characterize the two test cases are listed in Table 5.5.

Benchmark Quantities

In order to compare the computations quantitatively, three benchmark quantities are defined in [57]:

Test case	ρ_1	ρ_2	μ_1	μ_2	σ	Re	Eo	ρ_1/ρ_2	μ_1/μ_2
Elliptic	1000	100	10	1	24.50	35	10	10	10
Skirted ellipsoidal-cap	1000	1	10	0.1	1.96	35	125	1000	100

Table 5.5: Rising bubble - physical parameters and dimensionless numbers defining the test cases

- *Center of mass.* The center of mass $\boldsymbol{x}_c$ is defined by

$$\boldsymbol{x}_c = \begin{pmatrix} x_{1,c} \\ x_{2,c} \end{pmatrix} = \frac{\int_{\Omega_2} \boldsymbol{x} \, \mathrm{d}\boldsymbol{x}}{\int_{\Omega_2} 1 \, \mathrm{d}\boldsymbol{x}}.$$

  Due to the symmetry of the problem, only the vertical component $x_{2,c}$ of $\boldsymbol{x}_c$ is of interest.

- *Circularity.* The circularity c is defined as the ratio of the perimeter of an area-equivalent circle to the effective perimeter of the interface:

$$c = \frac{2(\pi \int_{\Omega_2} 1 \, \mathrm{d}\boldsymbol{x})^{1/2}}{\int_\Gamma 1 \, \mathrm{d}S}.$$

  A perfect circle has circularity 1, whereas every other shape has a lower circularity.

- *Rise Velocity.* The mean rise velocity $\boldsymbol{u}_c$ of the bubble is defined as

$$\boldsymbol{u}_c = \begin{pmatrix} u_{1,c} \\ u_{2,c} \end{pmatrix} = \frac{\int_{\Omega_2} \boldsymbol{u} \, \mathrm{d}\boldsymbol{x}}{\int_{\Omega_2} 1 \, \mathrm{d}\boldsymbol{x}}.$$

  Due to the symmetry of the problem, only the vertical component $u_{2,c}$ of $\boldsymbol{u}_c$ is of interest.

The line integral $\int_\Gamma 1 \, \mathrm{d}S$ for the circularity can be approximated by transforming it first to a volume integral involving a Dirac delta function which is then replaced by some regularization. If ϕ is the signed distance function, we have

$$\int_\Gamma 1 \, \mathrm{d}S = \int_\Omega \delta(\phi) \mathrm{d}\boldsymbol{x} \approx \int_\Omega \frac{1}{\varepsilon} \delta_r(\phi/\varepsilon) \mathrm{d}\boldsymbol{x}, \tag{5.1}$$

where $\delta(\cdot)$ is the one-dimensional Dirac delta function, $\delta_r(\cdot)$ is some regularization of δ, and ε is the regularization bandwidth, which is of the order of the mesh size h. If ϕ is not the signed distance function, the formula (5.1) has to be corrected:

$$\int_\Gamma 1 \, \mathrm{d}S \approx \int_\Omega \frac{|\boldsymbol{\nabla}\phi|}{\varepsilon} \delta_r(\phi/\varepsilon) \mathrm{d}\boldsymbol{x}, \tag{5.2}$$

which corresponds to (5.1) for the signed distance function, because $|\boldsymbol{\nabla}\phi| = 1$ in this case. Note that the line integral $\langle \sigma\kappa\boldsymbol{n}_\Gamma, \boldsymbol{v} \rangle$ in equation (2.20) for evaluating the surface tension can be evaluated using the same approach.
See [111] for an error analysis of this approach and concrete possibilities for the choice of δ_r.

Error Quantification

For better error assessment, the evolution of the benchmark quantities is measured against suitable reference solutions. Denoting the value of the benchmark quantity at time level t^n by q^n, the respective value of the reference solution by q^n_{ref} and the number of timesteps by NTS, we establish the following relative error norms [57]:

$$\|e(q)\|_1 = \frac{\sum_{n=1}^{NTS} |q^n_{ref} - q^n|}{\sum_{n=1}^{NTS} |q^n_{ref}|},$$
$$\|e(q)\|_2 = \left(\frac{\sum_{n=1}^{NTS} |q^n_{ref} - q^n|^2}{\sum_{n=1}^{NTS} |q^n_{ref}|^2} \right)^{1/2},$$
$$\|e(q)\|_\infty = \frac{\max_{n=1}^{NTS} |q^n_{ref} - q^n|}{\max_{n=1}^{NTS} |q^n_{ref}|},$$

where we use the numerical results of group 2 in [57] on the finest grid as reference solution, interpolated appropriately onto our time levels.
We will also compute estimated convergence rates ROC defined as

$$\mathrm{ROC}_p = \frac{\log(\|e^{l-1}\|_p / \|e^l\|_p)}{\log(h^{l-1}/h^l)},$$

where l is the grid refinement level.

5.2.2 Approximation Details

As we explained in Section 2.4, continuous interior penalty finite element formulations are not consistent in the presence of surface tension, because of the jump of the pressure at the interface. We therefore look at each timestep for numericals solutions of the flow equations $(\boldsymbol{u}_h(t^n), p_h(t^n))$ in the classical Taylor-Hood space X_h^2, using the Taylor-Hood scheme form Section 2.4, i.e., without any stabilization. The numerical solution of the level set equation $\phi_h(t^n)$ is searched for in V_h^1, using the interior penalty formulation from Section 2.6.
As iterative scheme, we use the fixpoint formulation from Subsection 3.1.4. The tolerance in the fixpoint iterations is fixed to $\varepsilon = 0.05$. This amounts in some way to restricting the error by nonlinearities to 5%.
Although we use an extrapolation of the level set function as initial guess when solving the fluid equations in the first fixpoint iteration, the treatment of the surface tension is of explicit character and imposes a constraint on the timestep. According to [9], it takes the form

$$\Delta t \leq C \left(\frac{(\rho_1 + \rho_2) h^3}{4\pi\sigma} \right)^{1/2}.$$

An implicit way of treating the surface tension has been proposed in [56] in order to circumvent this restriction.
As described in Subsection 3.5.3, we use interface local projection reinitialization combined with a fast marching method on the far field for reinitializing the level set function. We also apply the adaptive scheme described there to decide whether or not to reinitialize at a given time level. Note that the extrapolation of the level set function $\hat{\phi}^{n+1} = 2\phi^n - \phi^{n-1}$

used as initial guess in the fixpoint iteration is not consistent if ϕ^n has been replaced by its reinitialization. The extrapolation would somehow double (and thereby destroy) the effect of reinitialization. Also the consistency of the derivative approximation in the BDF2 scheme is not of second order any more. For this reason, the advection step of the level set function following a reinitialization is solved using a Crank-Nicolson time discretization instead. It uses only ϕ^n (which has been reinitialized) and not ϕ^{n-1}, and retains second order accuracy.

5.2.3 Results for Ellipsoidal Bubble

The first set of physical parameters leads to the development of an ellipsoidal bubble. The computations were performed on unstructured triangular meshes with $h \in \{0.1, 0.06, 0.04, 0.02\}$. Table 5.6 shows the simulation statistics for the different grid levels where the number of elements is denoted by NEL, the total number of degrees of freedom by NDOF (velocity, pressure and level set function), and the total number of time steps by NTS. The time in seconds required for each computation is denoted by CPU which, scaled by the number of time steps yields the factor CPU/TS.

h	NEL	NDOF	NTS	CPU	CPU/TS
0.1	408	2212	600	501	0.8
0.06	1180	6198	1500	4473	3.0
0.04	2600	13442	3000	20358	6.8
0.02	10386	52822	3000	136646	45.5

Table 5.6: Ellipsoidal bubble - simulation statistics

In Figure 5.14, the approximated bubble shape on the finest mesh at the final time $t = T = 3$ is shown. We can see that the bubble does deform from its initial circular shape, but it does not break up.

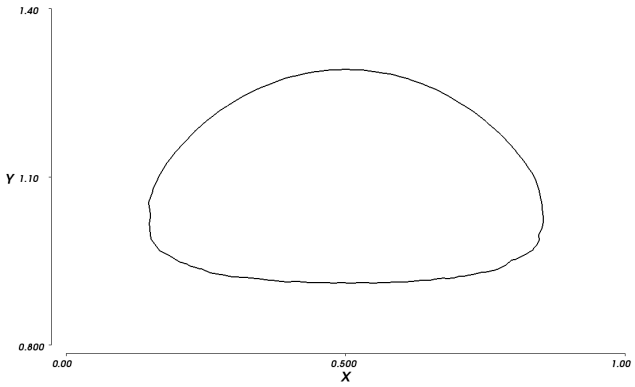


Figure 5.14: Ellipsoidal bubble - shape at time $t = 3$

The relative error norms for the circularity, center of mass, and rise velocity are shown in Table 5.7, together with the estimated convergence rates (ROC). The reference solution is taken as

the solution of the computation of the second group in [57] on their finest grid. The center

h	$\|e\|_1$	ROC_1	$\|e\|_2$	ROC_2	$\|e\|_\infty$	ROC_∞
Circularity c						
0.1	0.0294		0.0351		0.0922	
0.06	0.00976	2.16	0.0123	2.06	0.0298	2.21
0.04	0.00419	2.09	0.00501	2.21	0.0118	2.30
0.02	0.00224	0.91	0.00280	0.84	0.00765	0.62
Center of mass $x_{2,c}$						
0.1	0.0304		0.0379		0.0620	
0.06	0.0126	1.72	0.0166	1.62	0.0271	1.62
0.04	0.00440	2.59	0.00661	2.27	0.0133	1.76
0.02	0.00117	1.91	0.00162	2.03	0.00346	1.94
Rise velocity $u_{2,c}$						
0.1	0.113		0.140		0.224	
0.06	0.0636	1.13	0.0729	1.27	0.0939	1.71
0.04	0.0286	1.97	0.0340	1.88	0.0538	1.38
0.02	0.00533	2.42	0.00618	2.46	0.0148	1.86

Table 5.7: Ellipsoidal bubble - relative norms and convergence orders

of mass and the rise velocity approach or attain quadratic convergence in all three norms. Beside the last mesh, this also holds for the circularity. The decrease of convergence order for the last mesh can be explained by the fact that our scheme might converge (potentially with second order) to a slightly different solution than the scheme of group 2 in [57].
The following figures depict the time evolution of the benchmark quantities for different meshes. Figure 5.15 shows the evolution of the circularity. We can see a highly oscillating

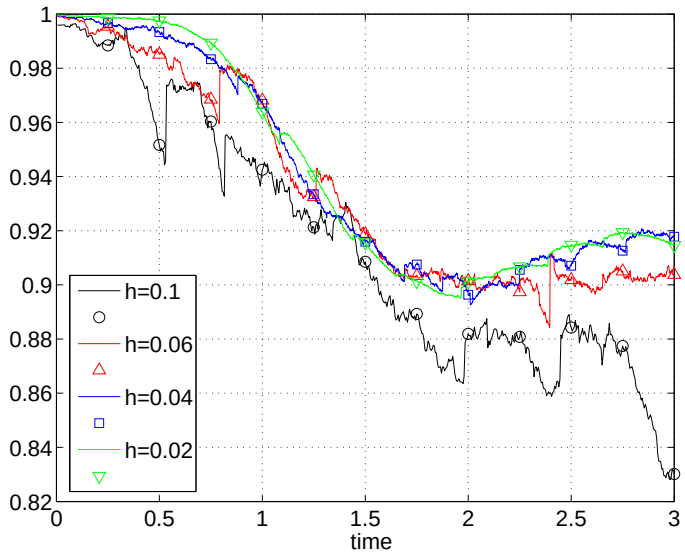


Figure 5.15: Ellipsoidal bubble - circularity

behaviour, which stems however from the computation of the circularity using equation (5.1) not taking into account that ϕ is not strictly a distance function, and not from the numerical solution itself. The biggest jumps in the curve of the coarsest mesh indicate the reinitialization times, where ϕ becomes a signed distance function again. The corrected method using equation (5.2) applied for the test case with lower surface tension does not show this behaviour (c.f. Figure 5.20).
Both the center of mass, shown in Figure 5.16, and the mean rise velocity, shown in Figure 5.17 converge very nicely.

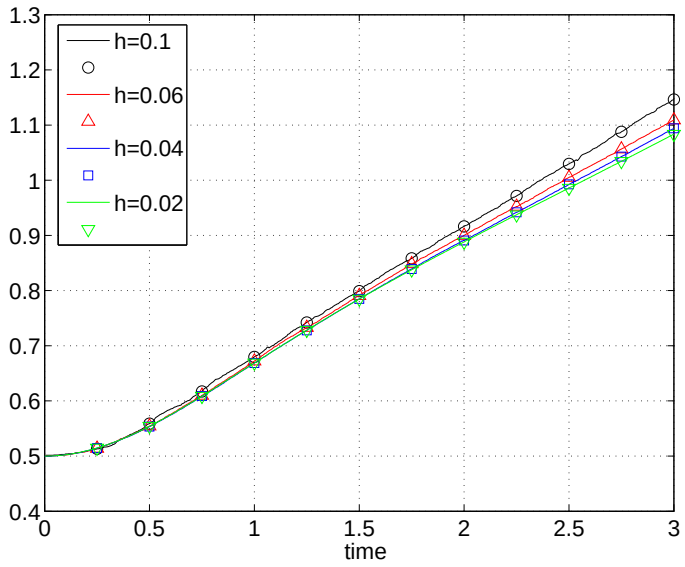


Figure 5.16: Ellipsoidal bubble - center of mass

Table 5.8 shows some extrema of the benchmark quantities along with their incidence times. All values on our finest mesh are close to but slightly outside the reference bounds, but this is natural because our meshes are still coarser than the ones used for establishing the bounds.

	$c_{\min}$	$t_{c_{\min}}$	$u_{c,2,\max}$	$t_{u_{c,2,\max}}$	$x_{c,2}(t=3)$
lower bound	0.9011	1.875	0.2417	0.921	1.081
upper bound	0.9013	1.905	0.2421	0.932	1.083
$h = 0.1$	0.8280	2.980	0.24844	1.150	1.1463
$h = 0.06$	0.8843	2.396	0.24848	0.978	1.1091
$h = 0.04$	0.8925	2.010	0.24317	0.916	1.0941
$h = 0.02$	0.8950	1.964	0.24166	0.949	1.0836

Table 5.8: Ellipsoidal bubble - extrema of benchmark quantities and their incidence times on different meshes, with bounds from [57]

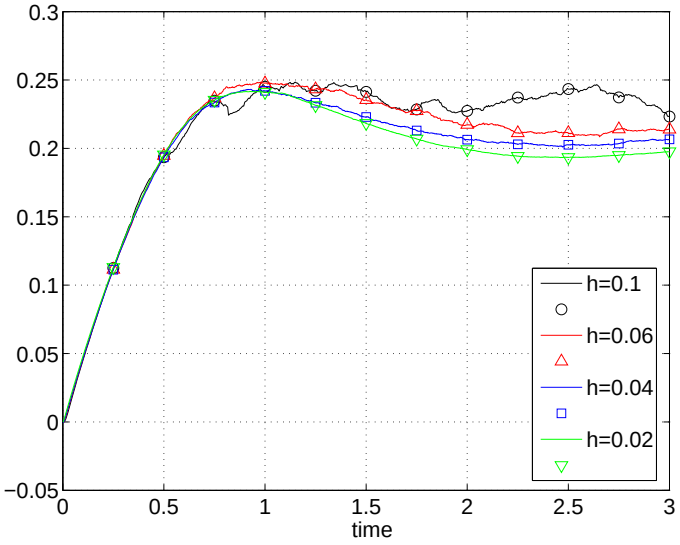
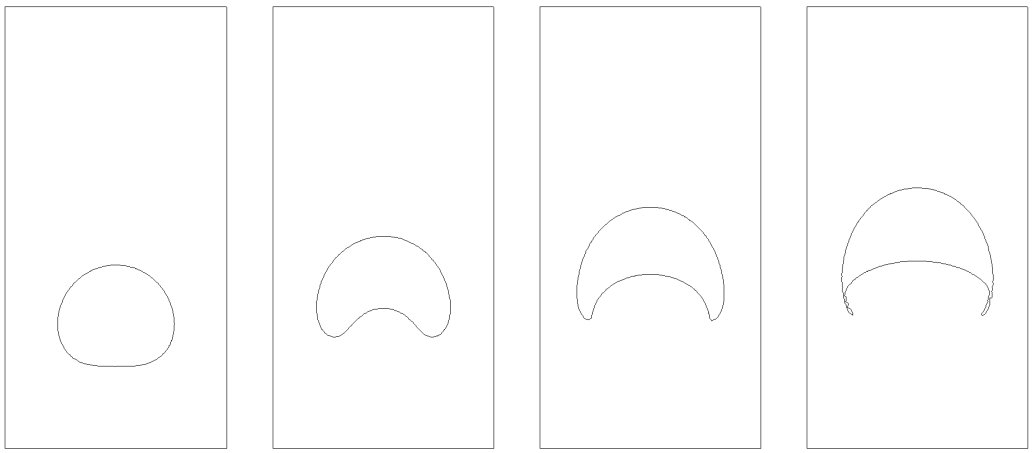


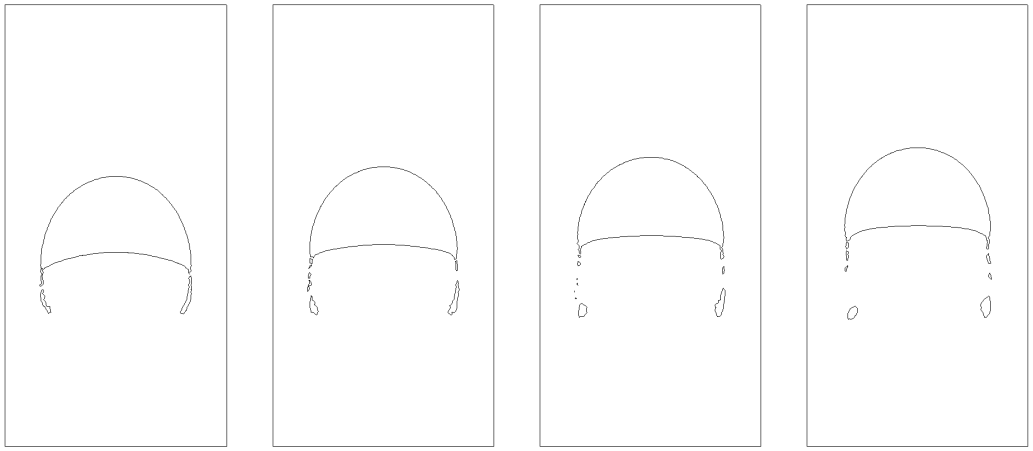
Figure 5.17: Ellipsoidal bubble - rise velocity

5.2.4 Results for Skirted Ellipsoidal-Cap Bubble

Figure 5.18 shows the evolution of the bubble interface, computed on our finest grid. It rises at about the same speed as the ellipsoidal bubble with the first set of physical parameters, but the lower surface tension causes this bubble to deform more. Thin filaments (skirts) develop and eventually break off. The time of break up is predicted to occur between $t = 2.2$ and $t = 2.4$, which is confirmed by the reference computations.
The computations were performed on unstructured triangular meshes with $h \in \{0.05, 0.025, 0.0125\}$. The bubble shapes at the final time $t = 3$ of our numerical solution are shown in Figure 5.19. We can see that the filaments on the coarsest mesh are too coarse as well, making the remaining part too small. This remaining ellipsoidal-cap bubble converges well, whereas this cannot be affirmed for the actual shape of the filaments. Further mesh refinement would be necessary to find a converged solution for the filament shape.
The curves for the circularity in Figure 5.20 show a typical convergence behaviour up to $t \approx 2$ after which the bubble breaks up and no convergence trend can be observed any more. Since the filaments do not retract after the breakup, the circularity continues to decrease.
The vertical position of the center of mass, shown in Figure 5.21, behaves like the circularity. We can see a typical convergence behaviour up to $t \approx 2$ after which the bubble breaks up and no convergence trend can be observed any more.
The same behaviour can also be observed for the mean rise velocity, which now shows two maxima, see Figure 5.22. Good convergence is observed until about the local minimum at $t \approx 1.5$. Then the curves start disconnecting, and the second maximum occurs later and later the finer the mesh becomes.
Table 5.9 shows some extrema of the benchmark quantities along with their incidence times. The two velocity maxima on our finest grid are very close to the span of the reference results, all other values are within the interval of results from [57]. Note however that some of these intervals are big, i.e., the codes do not agree. This is quite natural because already the physical stability of the problem is a delicate one.



(a) $t = 0.6$ (b) $t = 1.2$ (c) $t = 1.8$ (d) $t = 2.2$

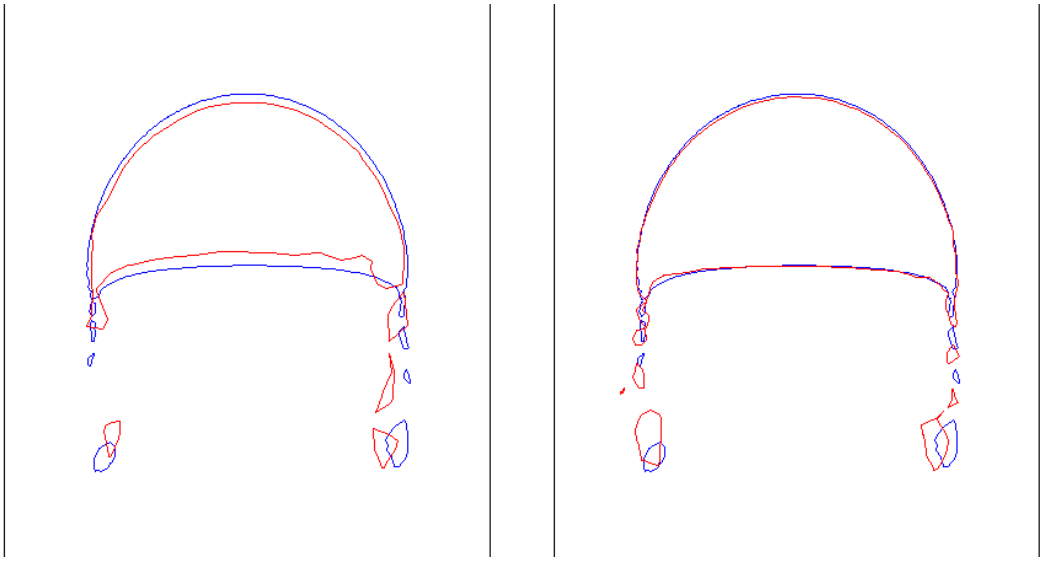


(e) $t = 2.4$ (f) $t = 2.6$ (g) $t = 2.8$ (h) $t = 3.0$

Figure 5.18: Skirted ellipsoidal-cap bubble - typical time evolution of interface

	$c_{\min}$	$t_{c_{\min}}$	$u_{c,2,\max 1}$	$t_{u_{c,2,\max 1}}$	$u_{c,2,\max 2}$	$t_{u_{c,2,\max 2}}$	$x_{c,2}(t=3)$
Group 1	0.5869	2.4004	0.2524	0.7332	0.2434	2.0705	1.1380
Group 2	0.4647	3.0000	0.2514	0.7281	0.2440	1.9844	1.1249
Group 3	0.5144	3.0000	0.2502	0.7317	0.2393	2.0600	1.1376
$h = 0.05$	0.5684	3.0000	0.2472	0.7400	0.2281	1.8150	1.1232
$h = 0.025$	0.5402	2.9800	0.2484	0.7350	0.2364	1.9950	1.1105
$h = 0.0125$	0.5853	2.7250	0.2495	0.7300	0.2382	2.0150	1.1254

Table 5.9: Skirted ellipsoidal-cap bubble - extrema of benchmark quantities and their incidence times on different meshes, with reference results of 3 codes from [57]



(a) $h = 0.05$ (b) $h = 0.025$

Figure 5.19: Skirted ellipsoidal-cap bubble - bubble shape at time $t = 3$. Coarse grid solutions (shown in red) compared to the shape computed on the finest grid $h = 0.0125$ (shown in blue).

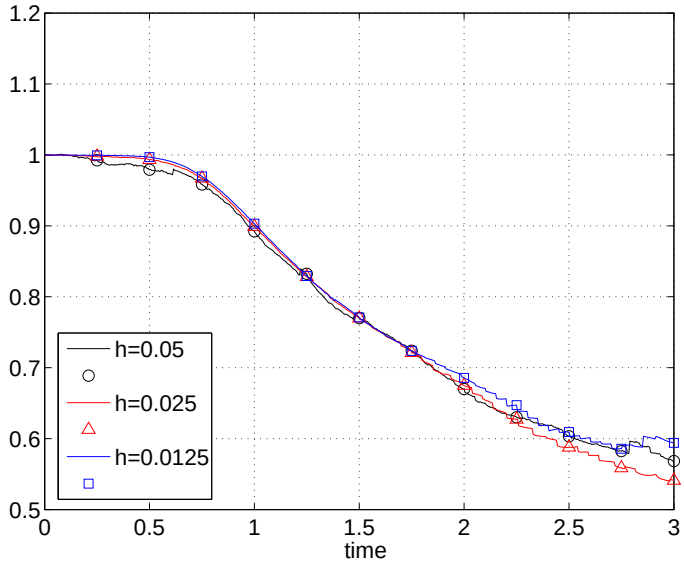


Figure 5.20: Skirted ellipsoidal-cap bubble - circularity

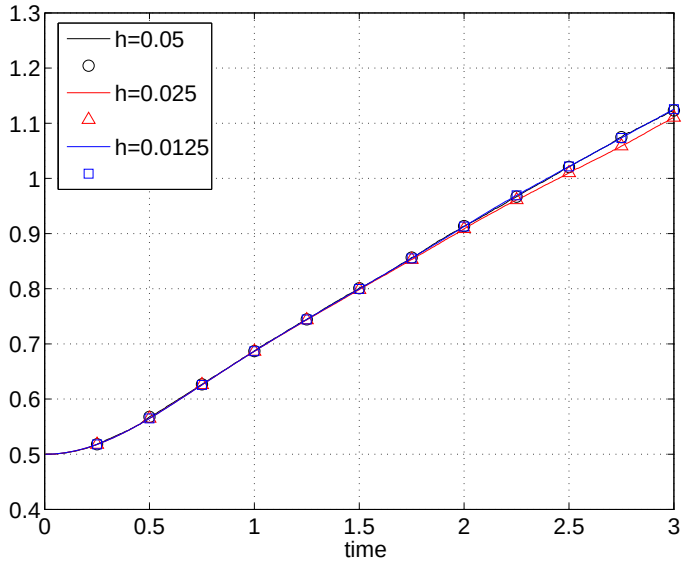


Figure 5.21: Skirted ellipsoidal-cap bubble - center of mass

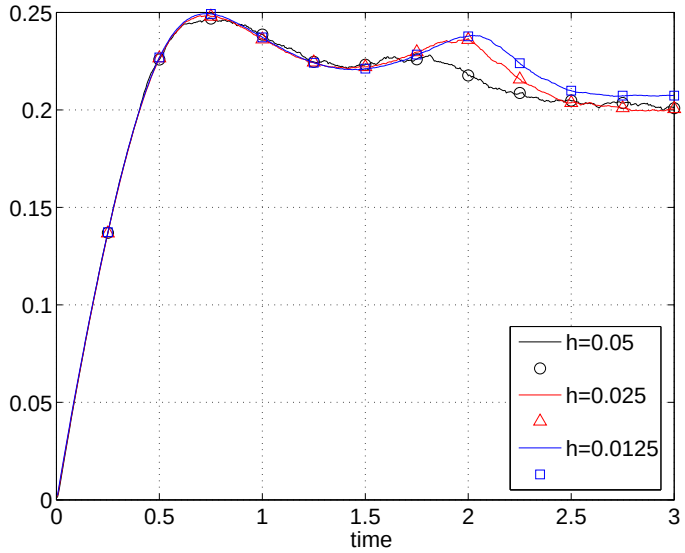


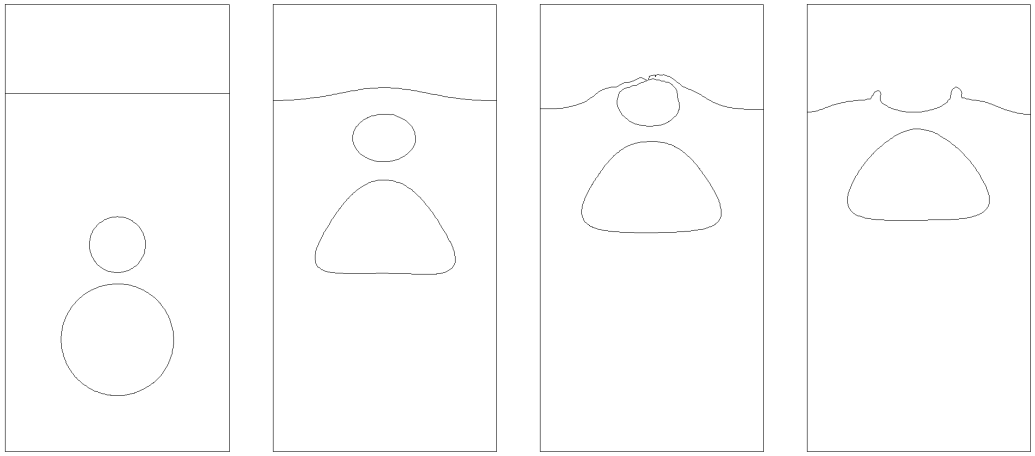
Figure 5.22: Skirted ellipsoidal-cap bubble - rise velocity

We can conclude that our implementation provides results that are in good agreement with other codes, as long as there is agreement at all.

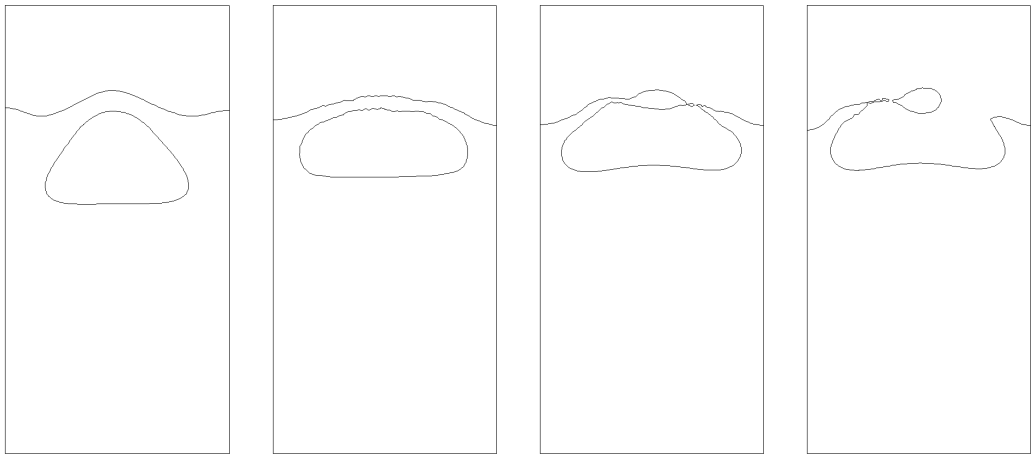
5.3 Two Rising Bubbles Undergoing Topology Change

In order to test and illustrate the capabilities of our approach to handle interfaces with changing topology, we carry out a simulation of two rising bubbles. We choose the domain, boundary conditions and the physical parameters of the ellipsoidal bubble from Section 5.2. But we add a smaller second bubble above the first one, and a free surface near the top of the domain (see Figure 5.23(a)). In order to see the two bubbles reach the surface and the latter re-equilibrate, we simulate until $T = 8$. Because bigger bubbles rise faster than small ones when they are not interacting, one could expect the two bubbles to merge before they join the surface.
Figure 5.23 shows the time evolution of the numerical solution. At $t = 2.0$, we see that the two bubbles have not merged. The distance between the two has rather increased, compared to the initial condition. It seems that the velocity field induced by the lower, bigger bubble pushes the small bubble to rise faster, avoiding the merger of the two bubbles. The free surface starts to build a bump. At $t = 2.7$, the small bubble has reached the surface and the small filament between them starts splitting up. The filament then retracts to the surface, as can be seen in Figure 5.23(d) for $t = 2.9$. As a consequence of this first bubble breakup, the free surface starts oscillating, deforming thereby the larger, remaining bubble. So it is almost triangular at $t = 3.2$, but becomes flattened at $t = 3.7$. The filament formed again starts splitting up at $t = 4$. Because the second bubble is bigger, the filament splits not once but twice, leaving not only two ends retracting to the free surface, but also a piece of fluid alone, see Figure 5.23(h). Due to the presence of surface tension, this rest of the filament takes the minimum energy shape of a circle while the two other ends retract ($t = 4.5$). This droplet now falls towards the oscillating free surface, touching it at about $t = 4.8$. The droplet merges, and it remains an oscillating free surface. This oscillation is damped by the surface tension as well as by the viscous effects in the two fluids.
Although the domain, the boundary and the initial conditions are perfectly symmetric, the numerical solution is not. This phenomenon is explained by the use of unstructured unsymmetric meshes which introduce small perturbations of the symmetry. These perturbations are then amplified at the breakup by the cohesive character of the two phases modeled by the surface tension.

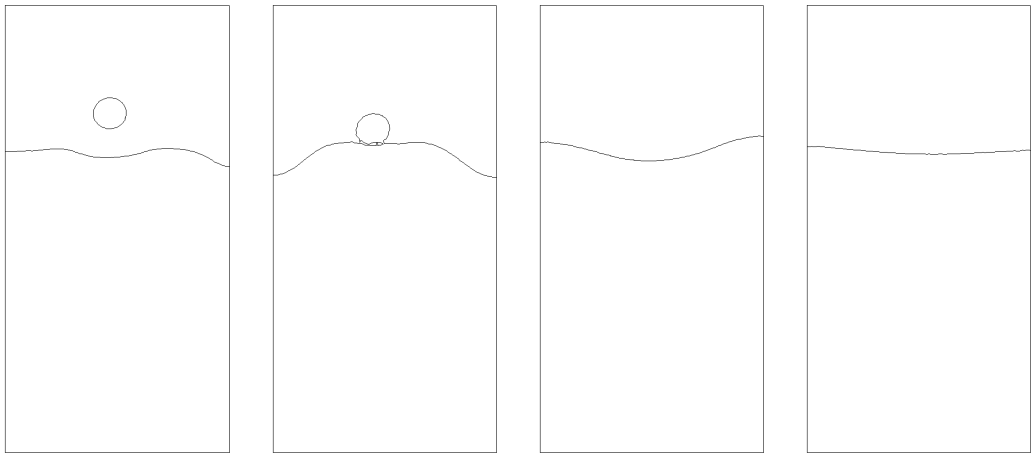
We can conclude that interior penalty stabilized finite element methods work well for the level set advection problem. Combined with suitable reinitialization techniques, our method reliably reproduces also difficult benchmark results of two fluid flow. Complex topology changes of the surface are handled without special intervention.



(a) $t = 0.0$ (b) $t = 2.0$ (c) $t = 2.7$ (d) $t = 2.9$



(e) $t = 3.2$ (f) $t = 3.7$ (g) $t = 4.0$ (h) $t = 4.1$



(i) $t = 4.5$ (j) $t = 4.8$ (k) $t = 6.0$ (l) $t = 8.0$

Figure 5.23: Two bubbles - time evolution of interface

Chapter 6

Software Design Aspects

In this chapter, we shortly present the software environment *Life*, in which the methods and algorithms from Chapters 2 and 3 were implemented and with which the results in Chapters 4 and 5 were obtained. Section 6.1 gives an overview of the project evolution and some details about the new mathematical kernel. In Section 6.2, we discuss the contributions added to the project in the framework of the present thesis.

6.1 The Life Project

The project *Life*, whose name stands for LIbrary of Finite Elements, has its origins in an early version in Fortran77 produced at the *Politecnico di Milano* (Italy) by Fausto Saleri and was then developed in a collaborative effort of researchers at *Politecnico di Milano*, *INRIA Rocquencourt* (France), and *École Polytechnique Fédérale de Lausanne* (Switzerland). A brief history of the project is sketched in Subsection 6.1.1, and the main ideas and features of the new mathematical kernel are listed in Subsection 6.1.2.

6.1.1 Project History

The first version of Life was developed in Fortran77, and the second version in Fortran90/95. These versions were essentially two dimensional. Based on the second version, N. Parolini implemented a two dimensional level set method for two fluid flow [84], including interface local projection reinitialization [21]. This code called *FreeLIFE* is one of the three codes used in [57] for establishing the benchmark computations of the rising bubble we are comparing our results to in Section 5.2.
Later, a three dimensional version, implemented in C++ and called *LifeV*, has been developed. It features piecewise constant, linear, bilinear and quadratic finite elements, as well as the $\mathbb{P}_1$-bubble element and the Raviart-Thomas element. LifeV has served as a testbed for the implementation of Navier-Stokes solvers with algebraic splitting (see e.g. [90]) and with interior penalty stabilization (see [17]). Based on these solvers, state of the art numerical methods for fluid-structure interaction have been implemented in LifeV (see e.g. [30]). Also, an expression template language based on LifeV was proposed in [32]. All three dimensional computations in this thesis were carried out using LifeV.
More recently, the mathematical kernel of LifeV grown organically has been completely reimplemented in C++ with more generality and expressivity [88], using state-of-the-art techniques like expression templates and metaprogramming. All two dimensional computations in this

thesis were carried out using this new building block. Its main features are presented in the following subsection.

6.1.2 New Mathematical Kernel

In the need for new features in LifeV, C. Prud'homme proposes a new mathematical kernel for the library, simply called *Life* again, see [88]. Its key ideas are:

- **Efficiency:** The implementation is careful about efficiency issues, and its design targets parallel computer architectures inherently.
- **Genericity:** Class templates allow for various numerical types with different (potentially arbitrary) precisions.
- **Abstraction:** The kernel follows closely the mathematical abstractions, allowing for an independent and general implementation of new methods and algorithms.

The main features of this new platform are:

- **Dimension:** uniform problem formulation in 1, 2, or 3 space dimensions
- **Point sets:** many point sets on the reference elements like Fekete and Gauss points for the definition of interpolants, quadrature methods and nodal bases of polynomial spaces
- **Polynomials:** hierarchic primal bases (moment basis, or L^2-orthonormal basis for better numeric stability) of any order on the reference elements, allowing efficient extraction of bases spanning lower order subspaces, exact integration, etc.
- **Function spaces:** continuous and discontinuous function space classes and classes for actual functions in these spaces
- **Expressions:** embedded language allowing efficient and succint definition of expressions involving finite element functions, analytic and algebraic operations
- **Interpolation:** interpolation of arbitrary expressions into a given function space
- **Integration:** numerical evaluation of integrals of arbitrary expressions over domains, boundaries or marked geometrical entities in general
- **Variational Formulation:** assembly of vectors and matrices representing linear and bilinear forms from integrals of arbitrary expressions over domains, boundaries or marked geometrical entities in general
- **Linear Algebra:** general interface for vector and matrix formats

The equivalence of mathematical formulation and its implementation can easily be seen in the following example: We want to assembly the matrix $\mathbf{A}$ representing the bilinear form

$$\begin{aligned} a : V_h^k \times V_h^k &\to \mathbb{R} \\ (u, v) &\to \sum_{K \in \mathcal{T}} \int_K \boldsymbol{\nabla} u \cdot \boldsymbol{\nabla} v \, \mathrm{d}\boldsymbol{x} \end{aligned}$$

in d dimensions. Listing 6.1 shows how to define the mesh $\mathcal{T}$, the function space V_h^k, the functions $u, v \in V_h^k$, the matrix $\mathbf{A}$ and the bilinear form a in the variational formulation language of Life.

Listing 6.1: Bilinear form example

```
// T_h
mesh_type mesh;

// {φ_i}_i : V_h^k = span_i φ_i
const int Dim = 3; // spatial dimension
const int k = 2; // polynomial degree
typedef fusion::vector<fem::Lagrange<Dim, k, Scalar,
                                     Continuous, double,
                                     Simplex> > basis_type;

// V_h^k
typedef FunctionSpace<mesh_type, basis_type, double> space_type;
space_type Vh(mesh);

// u, v ∈ V_h^k
space_type::element_type u(Vh), v(Vh);

// A = a(φ_j, φ_i);
// a(u,v) = Σ_{K∈T} ∫_K ∇u · ∇v dx
csr_matrix_type A;
form(Vh,Vh,A) = integrate( elements(mesh),
                           IM<Dim, 2*k-2, double, Simplex>(),
                           gradt(u)*trans(grad(v)) );
```

See [87] for a detailed description of the construction of the embedded language and the underlying concepts and ideas.

6.2 Contributions to Life

Based on the general framework of the new mathematical kernel of LifeV, we have implemented three mathematical abstractions particularly useful for the present thesis, yet sufficiently general to have a broad application range. In this section, we present the essential ideas of these contributions. The code listings provided are slightly simplified for illustrating these ideas. By consequence the cannot serve as a tutorial for the application of the software presented.

6.2.1 Linear Algebra Backend Abstraction

The Life mathematical kernel provides wrapper classes for vector and matrix types of several commonly used linear algebra packages like PETSc [2] or Trilinos [50]. These wrapper classes provide *generic* interfaces that allow the linear and bilinear forms to assemble vectors and

matrices in the same way for all linear algebra packages. The linear systems defined by these systems however have to be solved using the commands *specific* to the respective linear algebra package.
In order to improve this situation, we introduce the notion of a *linear algebra backend*. A linear algebra backend is a class that bundles together the definition of a vector and a matrix wrapper type, together with operations on these types, suitable for a given linear algebra package. In the terminology of *design patterns* [39], the linear algebra backend abstraction is an application of the facade pattern. The operations to be provided by each backend implementation are the matrix-vector multiplication, a linear solve with a given matrix and right hand side, and the dot product of two vectors. Moreover, applying the factory method design pattern, two methods for creating new vectors and matrices are provided in case the construction needs arguments specific to the linear algebra package. This is of particular interest in the context of parallel computing, where the matrices and vectors are distributed over several processors and their partitioning needs to be accounted for upon construction. The types and the operations have generic names and interfaces in the backend class, such that changing one backend for another, usually one line in the code, allows to change the linear algebra package completely. The minimal interface of a linear algebra backend is given in Listing 6.2.
Another advantage is that the adaptive strategy for reusing a preconditioner presented in Subsection 3.2.3 can be implemented as a wrapper backend around another backend. Using the decorator design pattern, the implementation looks like shown in Listing 6.3. With the *generic programming* paradigm, i.e. specifying the underlying backend as template parameter, the strategy has to be implemented only once and it immediately becomes available for every backend that satisfies the minimal interface from listing 6.2. Using *object orientation*, i.e. deriving the adaptive backend class from the underlying backend class, only the behaviour which changes has to be modified, all other features of the underlying backend are *inherited* by the adaptive backend.
This abstraction allows to write programs that are to a large extent independent of the linear algebra package to be used, and it also allows generic high level strategies like the adaptive preconditioner reuse strategy to be implemented in a way which is independent of the linear algebra package used. See Listing 6.4 for a simple example, where the dependence on the linear algebra package is reduced to one line.

6.2.2 Linear Functional and Operator Abstraction

The concept of a (linear) operator on a function space is widely used in mathematical formulations of variational problems. Inverses, adjoints, compositions, linear combinations and other operations defined on operators play an essential role in problem and algorithm formulations. Similar observations hold for linear functionals on function spaces.
In order to be able to formulate problems and solution strategies with the same expressivity in software as on the paper, we introduce the abstraction of operators and linear functionals in the software.
Let us first analyze the mathematical situation for clarity. If we consider a linear form l on some (discrete) function space V, i.e., $l : V \to \mathbb{R}$, we may write

$$l(v) = \langle L, v \rangle = \boldsymbol{V}^\top \boldsymbol{L}, \tag{6.1}$$

Listing 6.2: Minimal interface of a linear algebra backend

```
class ExampleBackend
{
public:
  // type definitions
    typedef ExampleMatrix sparse_matrix_type;
    typedef ExampleVector vector_type;
    typedef double value_type;

  // default constructor
    ExampleBackend();

  // factory methods
    template<typename DomainSpace, typename DualImageSpace>
    static sparse_matrix_type* newMatrix(DomainSpace const&,
                                         DualImageSpace const&);
    template<typename SpaceT>
    static vector_type* newVector( SpaceT const& space );

  // linear algebra interface
    template <class Vector>
    static void applyMatrix( sparse_matrix_type const& A,
                             const Vector& x,
                             vector_type& b );
    template <class Vector>
    void solve( sparse_matrix_type const& A,
                Vector& x,
                const vector_type& b,
                bool reusePC = false );
    template <class Vector>
    static value_type dot( const vector_type& f,
                           const Vector& x );

  // getting details about the solver
    bool converged();          // if the solver converged
    size_type get_iteration(); // number of iterations
}; // class ExampleBackend
```

Listing 6.3: Definition of the backend class implementing the adaptive reuse strategy from Subsection 3.2.3 around any other backend given as template argument

```
template<class Backend>
class BackendAdaptiveReusePC : public Backend
{
public:
  // type definitions (forwarded)
    typedef Backend backend_type;
    // other type definitions inherited

  // modify solve method from linear algebra interface
  // if reusePC specified, bypass adaptive behaviour
    template <class Vector>
    inline void solve( sparse_matrix_type const& A,
                       Vector& x,
                       const vector_type& b,
                       bool reusePC )
    { ((backend_type*)this)->solve( A, x, b, reusePC ); }

  // if reusePC not specified, use adaptive version
    template <class Vector>
    void solve( sparse_matrix_type const& A,
                Vector& x,
                const vector_type& b )
    {
        ((backend_type*)this)->solve( A, x, b, M_reusePC );
        // decide if preconditioner should be reused next time
        // and store result in M_reusePC
    }

  // getting more details about the solver
    inline bool reusePC() const;     // if the PC will be reused
    inline bool reusedPC() const;   // if the PC has been reused
    inline bool reuseFailed() const; // if the PC reuse failed
}; // class BackendAdaptiveReusePC
```

Listing 6.4: Example of the application of backends

```
// definition of the backend type
// the only line specific to the linear algebra package used
typedef BackendAdaptiveReusePC<ExampleBackend> backend_type;

backend_type theBackend;
backend_type::sparse_matrix_type A;
backend_type::vector_type b;
backend_type::vector_type x;

// fill the matrix A and the right hand side vector b here

// solve for x in Ax = b
theBackend.solve( A, x, b );
```

where $\langle \cdot, \cdot \rangle$ denotes the duality pairing and the vectors $\boldsymbol{V}$ and $\boldsymbol{L}$ are such that

$$(\boldsymbol{L})_i = l(\varphi_i) = \langle L, \varphi_i \rangle, \quad i = 1 \dots M, \qquad \text{and} \qquad v(\boldsymbol{x}) = \sum_{i=1}^{M} (\boldsymbol{V})_i \; \varphi_i(\boldsymbol{x})$$

for some basis $\{\varphi_i\}_{i=1}^{M}$ of V. It is clear that $L \in V'$, the dual space of V. The distinction between the linear form l and the linear functional L is of course somewhat artificial.
If we now consider a bilinear form $a : W \times V \to \mathbb{R}$, we may write analogously

$$a(u, v) = \langle Au, v \rangle = \boldsymbol{V}^\top \mathbf{A} \boldsymbol{U}, \tag{6.2}$$

where $\boldsymbol{V}$ is as above and $\boldsymbol{U}$ and $\mathbf{A}$ are such that

$$(\mathbf{A})_{ij} = a(\psi_j, \varphi_i) \quad i = 1 \dots M, \qquad \text{and} \qquad u(\boldsymbol{x}) = \sum_{j=1}^{N} (\boldsymbol{U})_j \; \psi_j(\boldsymbol{x})$$

for some bases $\{\varphi_i\}_{i=1}^{M}$ of V and $\{\psi_j\}_{j=1}^{N}$ of W.
Note that we have three mathematically distinct objects: the bilinear form a, the operator A and the matrix $\mathbf{A}$. Comparing equations (6.1) and (6.2), we can see that A is a linear operator from W to V', and that Au is a linear functional represented by a vector obtained from the multiplication $\mathbf{A}\boldsymbol{U}$.
In Life, there are already classes representing the function spaces V and W, their elements v and u, but also the forms a and l. Defining linear functionals and operators in this framework means just closing a small gap. Using the paradigm of generic programming, the linear algebra backends introduced in Subsection 6.2.1 is passed as template parameter of the classes for the linear functionals and the linear operators. Their internal representation uses then vector and matrix types defined by the backend type, whereas their implementation is independent of this aspect. Also the application of a linear functional will use the dot product from the backend, the application of a linear operator uses the matrix vector product, and the application of its inverse will use the linear solving method from the backend. The introduction of classes for linear functionals and operators thereby reduces the dependence of the code on the linear

algebra package again. The notation allows furthermore to distinguish vectors representing functions in V from vectors representing functionals in V'.
We consider as an example the L^2 projection of the function $\sin(2x_2)$ into the discrete function space V_h^k and the computation of the L^2 norm of the projected discrete function u. More formally, this problem is written as follows:
Find $u \in V_h$ such that

$$\langle Mu, v\rangle = \langle L, v\rangle, \quad \forall v \in V_h,$$

where

$$\langle Mu, v\rangle = (u, v) \qquad \text{and} \qquad \langle L, v\rangle = (\sin(2x_2), v).$$

So we have formally $Mu = L$ and thus $u = M^{-1}L$. The L^2-norm of u is

$$\|u\|_{0,\Omega} = \sqrt{(u,u)} = \sqrt{\langle Mu, u\rangle} = \sqrt{(Mu)(u)},$$

remembering that $Mu \in V_h'$.
Listing 6.5 shows the one to one correspondence between the mathematical formulation above and the formulation in the code. Note that all linear algebra representations and operations are entirely hidden to the user.

6.2.3 Oseen Problem as Versatile Framework

In Subsection 2.1.2, we transformed the time-dependent Navier-Stokes equations to an Oseen problem by discretizing in time and subsequent linearization. The objective of this approach is twofold. On one side, it allows to apply theoretical results obtained for the Oseen problem, or, seen from the other side, to analyze a whole class of problems at once. On the other side, it also permits to use the same implementation for all problems that can be brought to the form of equations (2.4)-(2.5), recalled here for reference:

$$\begin{aligned} \alpha\boldsymbol{u} + (\boldsymbol{\beta}\cdot\boldsymbol{\nabla})\boldsymbol{u} - \boldsymbol{\nabla}\cdot(2\mu\mathbf{D}(\boldsymbol{u})) + \boldsymbol{\nabla}\, p &= \boldsymbol{f} \quad \text{in } \Omega, \\ \boldsymbol{\nabla}\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega. \end{aligned}$$

The parametric data of this problem are α, $\boldsymbol{\beta}$, μ and $\boldsymbol{f}$. Different choices of these parameters allow to model many different problems:

- stationary problems with $\alpha = 0$,
- time-dependent problems with α and $\boldsymbol{f}$ from time discretization,
- classical Navier-Stokes problems with $\boldsymbol{\beta} = \hat{\boldsymbol{u}}$, some approximation of $\boldsymbol{u}$, μ =constant,
- Euler problems with $\boldsymbol{\beta} = \hat{\boldsymbol{u}}$ and $\mu = 0$,
- Stokes problems with $\boldsymbol{\beta} = \mathbf{0}$,
- Darcy problems with $\boldsymbol{\beta} = \mathbf{0}$ and $\mu = 0$,
- arbitrary Lagrangian/Eulerian formulations (ALE) with $\boldsymbol{\beta} = \hat{\boldsymbol{u}} - \boldsymbol{w}$, where $\boldsymbol{w}$ is the mesh velocity,

Listing 6.5: Projection example using linear functionals and operators

```
// T_h
mesh_type mesh;

// {φ_i}_i : V_h^k = span_i φ_i
const int Dim = 3; // spatial dimension
const int k = 2; // polynomial degree
typedef fusion::vector<fem::Lagrange<Dim, k, Scalar,
                                     Continuous, double,
                                     Simplex> > basis_type;

// V_h^k
typedef FunctionSpace<mesh_type, basis_type, double> space_type;
space_type Vh(mesh);

// u, v ∈ V_h^k
space_type::element_type u(Vh), v(Vh);

// linear algebra backend
typedef BackendGmm<double> backend_type;
backend_type backend;

// ⟨L, v⟩ = Σ_{K∈T} ∫_K sin(2x_2) v dx
LinearFunctional<space_type, backend_type> L(Vh);
L = integrate( elements(mesh),
               IM<Dim, 2*k, double, Simplex>(),
               sin(2*Py())*id(v) );

// ⟨Mu, v⟩ = Σ_{K∈T} ∫_K u v dx
LinearOperator<space_type, space_type, backend_type>
  M(Vh, Vh, backend);
M = integrate( elements(mesh),
               IM<Dim, 2*k, double, Simplex>(),
               idt(u)*id(v) );

// u = M^{-1} L
u = M.applyInverse(L);

// ‖u‖_{0,Ω} = sqrt((Mu)(u))
double norm = std::sqrt(M(u)(u));
```

- two fluid flows, where α and $\boldsymbol{\beta}$ depend on ρ, and ρ and μ depend on the level set function ϕ,
- and combinations of the above.

As long as the underlying numerical scheme is appropriate for the given parameter choices, one implementation of the Oseen problem may serve for the numerical approximation of all these problems. The most critical choice is $\mu = 0$ in the Euler and Darcy case. For continuous interior penalty formulations, these cases are analyzed in [19] for the Darcy problem, and in [16] for the Euler problem.
Using again generic programming techniques and the flexibility of Life, it is possible to implement a solver class for the Oseen problem, where the backend type and the underlying function spaces for velocity and pressure are template parameters. The expression types for the parametric data α, $\boldsymbol{\beta}$, μ and $\boldsymbol{f}$ in B_h (2.14) and f_h (2.20) are in their turn template parameters of the `update` member function of the Oseen class. Thereby, the implementation becomes independent of thes aspects and allows for a real abstraction.
In Listing 6.6, we show the core part of the interface of the Oseen Solver class. The arguments of the constructor of the class are: the finite element space for the velocities V_h^k, the finite element space for the pressure V_h^l, the linear algebra backend and two lists of flags describing the partition of the boundary into the mixed Robin part Γ_R and the Neumann part Γ_N. The first argument of the update function is an object that describes the set of elements S on which to integrate the incremental terms of reaction and diffusion, following the idea of incremental matrix update in Section 3.4. The following arguments of the function `update` are the expressions for the parametric data α, $\boldsymbol{\beta}$, μ and $\boldsymbol{f}$, for the boundary conditions functions $\boldsymbol{g}_D$, $\boldsymbol{g}_N$ and for the parameter ω in the mixed Robin boundary condition. Note that α and μ are needed as incremental expressions for the incremental matrix update. The last parameter tells the update function whether to update also the stabilization matrix. Following the ideas of Subsection 3.3.1, this is reasonably the case when the preconditioner is not going to be reused. This value will therefore come from the function `reusePC` of the backend for the adaptive preconditioner strategy (see Listing 6.3).

Benefits

Using the same code for all these problems has several benefits. The most obvious one is of course the reduced need of repeating almost identical implementations. As a consequence, all applications of the code benefit from error corrections, improvements and new features immediately. This unified approach also allows to test the general implementation thoroughly with simple problems, using it without change for more complex problems later. The simple stationary Kovasznay flow from Subsection 4.1.1 for example has been carried out with the same Oseen code as the two bubble simulation in Section 5.3. This makes sure that newly occuring problems when considering more difficult equations stem from the additional complexity and not from the introduction of a basic error due to reimplementation of the flow code. The unified treatment of polynomial orders, numerical types and dimension of the domain in Life broadens again the application range of such an implementation.

Efficiency Pitfall and Solutions

The main concern of such a general implementation are of course potential efficiency problems. Generality may introduce a severe computational overhead if not handled carefully. Three

Listing 6.6: Interface of the Oseen solver class (core parts)

```
template<class Space_u,
         class Space_p,
         class Backend>
class Oseen
{
public:
  // constructor
    Oseen( const Space_u& spaceU, // V_h^k
           const Space_p& spaceP, // V_h^l
           const Backend& backend,
           const std::set<flag_type>& robinFlags,     // Γ_R
           const std::set<flag_type>& neumannFlags ); // Γ_N

  // update operator and right hand side with given expressions
  // using B_h and f_h
    template<typename ItRange, typename EalphaInc,
             typename EmuInc, typename EmuAbs,
             typename Ebeta, typename Ef, typename EgD,
             typename EgN, typename Eomega>
    void update( const ItRange&   itRange,     // S
                 const EalphaInc& alphaInc,    // α_δ
                 const EmuInc&    muInc,       // μ_δ
                 const EmuAbs&    muAbs,       // μ
                 const Ebeta&     beta,        // β
                 const Ef&        f,           // f
                 const EgD&       gD,          // g_D
                 const EgN&       gN,          // g_N
                 const Eomega&    omega,       // ω
                 bool  updateStabilization );

  // solve linear system using the specified backend
    void solve();

  // return results
    const Space_u::element_type& velocityX();
    const Space_u::element_type& velocityY();
    const Space_u::element_type& velocityZ();
    const Space_p::element_type& pressure();
}; // class Oseen
```

features of Life help us to keep the overhead low.

1. The first one is the use of expression templates for α, $\boldsymbol{\beta}$, μ and $\boldsymbol{f}$, which take a certain type at compile time. When μ is a simple constant, the code compiles to a considerably simpler program than for the case where μ is a nonlinear expression of the level set function ϕ, which is in turn a finite element function to be evaluated.

2. The second feature that helps to reduce the overhead even further is the concept of integration over subdomains. The reaction term and the viscous term are integrated over a subdomain that can be specified by the user, and the result is added to the previous integration. Thereby, the viscous and the reaction term can be integrated once in a classical Navier-Stokes problem, and in subsequent updates of the matrix for the modified convection term, an empty domain can be specified as subdomain for integrating the reaction and the viscous term. But this technique also allows for an easy implementation of the incremental matrix update presented in Section 3.4, where the subdomain is the union of elements where the signs of the level set function has changed.

3. The third feature is the special care for formulations that allow state of the art compilers to optimize away unnecessary code. Passing zero as expression for μ will not only eliminate the viscous term numerically, but it will actually eliminate big parts of the binary code for computing this term.

Using all these features, it has been possible to implement a general and efficient solver for many problems fitting in the framework of Oseen problems.

Conclusions

In this thesis we adressed the mathematical and numerical analysis of interior penalty finite element approximations of the Navier-Stokes equations and the mathematical and numerical modelling of incompressible free surface flows.

First, we described the two-phase flow problem and its mathematical models in a level-set formulation. For the case of the two dimensional stationary two-phase Stokes equations, we proposed an unfitted finite element scheme with interior penalty stabilization. For this scheme, we proved optimal a priori error estimates in the energy norm for both velocity and pressure. This scheme can be generalized to the time dependent Navier-Stokes equations. For the three dimensional time dependent two-phase Navier-Stokes equations and the level set advection problem, we formulated interior penalty finite element schemes.

The numerical problems that are obtained from the discretization of the two-fluid flow equations are computationally complex. This complexity calls for the development of appropriate algorithms.

We reviewed state-of-the art linear solver approaches for saddle point problems stemming from flow equations. Because of nonlinearity and time dependence of two fluid flows, sequences of linear systems with slowly varying matrices have to be solved. For this situation, we proposed a preconditioning strategy with adaptively reusable preconditioner.

For the efficient treatment of the interior penalty stabilization terms, we studied two complementary improvements. We proposed an effective algorithm for the reduction of the assembly cost of the stabilization matrix, and a splitting strategy for the reduction of its stencil and hence its memory requirements.

To reduce the increased cost of multiphase flow because of nonconstant mass and diffusion matrices, we introduced an incremental matrix update method. Still in the context of multiphase flow, a reformulation of the fast marching method for the level set reinitialization was introduced, providing an efficent implementation and geometric insight.

The methods and algorithms above have been tested on reference and benchmark problems in two and three dimensions, demonstrating their accuracy and efficiency. To this end, we have designed and implemented general and efficient abstractions in a finite element code, using state-of-the-art programming techniques.

Bibliography

[1] D. Adalsteinsson and J. A. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148(1):2–22, 1999.

[2] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[3] J. Becker. A second order backward difference method with variable steps for a parabolic problem. *BIT*, 38(4):644–662, 1998.

[4] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numer.*, 14:1–137, 2005.

[5] K. Blanckaert. *Flow and turbulence in sharp open-channel bends.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2002.

[6] H. Blank, M. Rudgyard, and A. Wathen. Stabilised finite element methods for steady incompressible flow. *Comput. Methods Appl. Mech. Engrg.*, 174(1-2):91–105, 1999.

[7] M. Braack. Solution of complex flow problems: mesh- and model adaptation with stabilized finite elements. Habilitation thesis, Universität Heidelberg, 2004.

[8] M. Braack and T. Richter. Solutions of 3D Navier-Stokes benchmark problems with adaptive finite elements. *Computers and Fluids*, 35(4):372–392, 2006.

[9] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100(2):335–354, 1992.

[10] S. C. Brenner. Poincaré-Friedrichs inequalities for piecewise H^1 functions. *SIAM J. Numer. Anal.*, 41(1):306–324 (electronic), 2003.

[11] S. C. Brenner. Korn's inequalities for piecewise H^1 vector fields. *Math. Comp.*, 73(247):1067–1087 (electronic), 2004.

[12] F. Brezzi and R. S. Falk. Stability of higher-order Hood-Taylor methods. *SIAM J. Numer. Anal.*, 28(3):581–590, 1991.

[13] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*, volume 15 of *Springer Series in Computational Mathematics*. Springer-Verlag, New York, 1991.

[14] F. Brezzi, L. P. Franca, T. J. R. Hughes, and A. Russo. $b = \int g$. *Comput. Methods Appl. Mech. Engrg.*, 145(3-4):329–339, 1997.

[15] E. Burman and M. Fernández. Time stepping schemes for CIP stabilization of finite element methods. Part II: Reducing the stencil. in preparation.

[16] E. Burman and M. A. Fernández. Continuous interior penalty finite element method for the time-dependent Navier-Stokes equations: space discretization and convergence. *Numerische Mathematik*, 107:39–77, 2007.

[17] E. Burman, M. A. Fernández, and P. Hansbo. Continuous interior penalty finite element method for Oseen's equations. *SIAM Journal on Numerical Analysis*, 44(3):1248–1274, 2006.

[18] E. Burman and P. Hansbo. Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(15-16):1437–1453, 2004.

[19] E. Burman and P. Hansbo. Edge stabilization for the generalized Stokes problem: A continuous interior penalty method. *Computer Methods in Applied Mechanics and Engineering*, 195(19-22):2393–410, 2006.

[20] E. Burman, P. Hansbo, and C. Winkelmann. An unfitted finite element method for incompressible heterogeneous linear elasticity. in preparation.

[21] E. Burman and N. Parolini. A new reinitialization procedure for the finite element approximation of the level set equation. Technical report, IACS-EPFL, 2005.

[22] A. Caboussat. *Analysis and numerical simulation of free surface flows.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2003.

[23] A. Caboussat. Numerical simulation of two-phase free surface flows. *Arch. Comput. Meth. Engng.*, 12(2):165–224, 2005.

[24] Z.-H. Cao. Fast iterative solution of stabilized Navier-Stokes systems. *Appl. Math. Comput.*, 157(1):219–241, 2004.

[25] Z.-H. Cao. Fast Uzawa algorithms for solving non-symmetric stabilized saddle point problems. *Numer. Linear Algebra Appl.*, 11(1):1–24, 2004.

[26] T. Chen, P. D. Minev, and K. Nandakumar. A projection scheme for incompressible multiphase flow using adaptive Eulerian grid. *Internat. J. Numer. Methods Fluids*, 45(1):1–19, 2004.

[27] J. Chessa and T. Belytschko. An enriched finite element method and level sets for axisymmetric two-phase flow with surface tension. *Internat. J. Numer. Methods Engrg.*, 58(13):2041–2064, 2003.

[28] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Proc. Cambridge Philos. Soc.*, 43:50–67, 1947.

[29] S. Deparis. *Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2004.

[30] S. Deparis, M. Discacciati, G. Fourestey, and A. Quarteroni. Fluid-structure algorithms based on Steklov-Poincaré operators. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5797–812, 2006.

[31] A. Dervieux and F. Thomasset. *Approximation Methods for Navier–Stokes Problems*, volume 771 of *Lecture Notes in Mathematics*, chapter A finite element method for the simulation of Rayleigh-Taylor instability, pages 145–158. Springer-Verlag, Berlin, 1980.

[32] D. Di Pietro and A. Veneziani. Expression templates implementation of the continuous and discontinuous galerkin methods. *Computing and Visualization in Science*, 2005. to appear.

[33] J.-J. Droux and T. J. R. Hughes. A boundary integral modification of the Galerkin least squares formulation for the Stokes problem. *Comput. Methods Appl. Mech. Engrg.*, 113(1-2):173–182, 1994.

[34] H. C. Elman, D. J. Silvester, and A. J. Wathen. Block preconditioners for the discrete incompressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 40:333–344, 2002.

[35] H. C. Elman, D. J. Silvester, and A. J. Wathen. Performance and analysis of saddle point preconditioners for the discrete steady-state Navier-Stokes equations. *Numer. Math.*, 90:665–588, 2002.

[36] C. R. Ethier and D. A. Steinman. Exact fully 3D Navier-Stokes solutions for benchmarking. *Int. J. Numer. Meth. Fluids*, 19:369–375, 1994.

[37] L. P. Franca and S. L. Frey. Stabilized finite element methods. II. The incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 99(2-3):209–233, 1992.

[38] S. Fujima and K. Ohmori. Benchmark problems for numerical schemes to passively transported interface. *Int. J. Comput. Fluid Dyn.*, 18(4):317–322, 2004.

[39] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[40] S. Ganesan, G. Matthies, and L. Tobiska. On spurious velocities in incompressible flow problems with interfaces. *Comput. Methods Appl. Mech. Engrg.*, 196(7):1193–1202, 2007.

[41] R. Glowinski and O. Pironneau. Finite element methods for Navier-Stokes equations. *Annu. Rev. Fluid Mech.*, 24:167–204, 1992.

[42] S. Gross and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. *Journal of Computational Physics*, 224(1):40–58, May 2007.

[43] J.-L. Guermond and L. Quartapelle. On stability and convergence of projection methods based on pressure Poisson equation. *Int. J. Numer. Meth. Fluids*, 26:1039–1053, 1998.

[44] J.-L. Guermond and J. Shen. A new class of truly consistent splitting schemes for incompressible flows. *J. Comput. Phys.*, 192(1):262–276, 2003.

[45] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput. Methods Appl. Mech. Engrg.*, 191(47-48):5537–5552, 2002.

[46] P. Hansbo and A. Szepessy. A velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations. *Comp. Meth. Appl. Mech. Engrg.*, 84:175–192, 1990.

[47] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, 8(12):2182–2189, 1965.

[48] Y. He. A fully discrete stabilized finite-element method for the time-dependent Navier-Stokes problem. *IMA J. Numer. Anal.*, 23(4):665–691, 2003.

[49] Y. He and W. Sun. Stabilized finite element method based on the Crank-Nicolson extrapolation scheme for the time-dependent Navier-Stokes equations. *Math. Comp.*, 76(257):115–136 (electronic), 2007.

[50] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley. An overview of the Trilinos Project. *ACM Trans. Math. Software*, 31(3):397–423, 2005.

[51] J. G. Heywood, R. Rannacher, and S. Turek. Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 22(5):325–352, 1996.

[52] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comp. Phys.*, 39:201–225, 1981.

[53] J. Hnat and J. Buckmaster. Spherical cap bubbles and skirt formation. *Phys. Fluids*, 19:162–194, 1976.

[54] D. L. Hu and J. W. M. Bush. Meniscus-climbing insects. *Nature*, 437:733–736, 2005.

[55] T. J. R. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics. V. Circumventing the Babuška-Brezzi condition: a stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations. *Comput. Methods Appl. Mech. Engrg.*, 59(1):85–99, 1986.

[56] S. Hysing. A new implicit surface tension implementation for interfacial flows. *Internat. J. Numer. Methods Fluids*, 51(6):659–672, 2006.

[57] S. Hysing, S. Turek, D. Kuzmin, N. Parolini, E. Burman, S. Ganesan, and L. Tobiska. Proposal for quantitative benchmark computations of bubble dynamics. *International Journal for Numerical Methods in Fluids*, 2007. Submitted.

[58] M. Joerg. Numerical investigations of wall boundary conditions for two-fluid flows. Master's thesis, École Polytechnique Fédérale de Lausanne, 2005.

[59] V. John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3D Navier-Stokes equations. *Internat. J. Numer. Methods Fluids*, 40(6):775–798, 2002.

[60] V. John, G. Matthies, and J. Rang. A comparison of time-discretization/linearization approaches for the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 195(44-47):5995–6010, 2006.

[61] G. Jothiprasad, D. J. Mavriplis, and D. A. Caughey. Higher-order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes. *Journal of Computational Physics*, 191(2):542–566, 2003.

[62] H. Kajitani, H. Miyata, M. Ikehata, H. Tanaka, H. Adachi, M. Namimatzu, and S. Ogiwara. Summary of the cooperative experiment on Wigley parabolic model in Japan. In *Proc. of the 2nd DTNSRDC Workshop on Ship Wave Resistance Computations (Bethesda, USA)*, pages 5–35, 1983.

[63] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *J. Comput. Phys.*, 59(2):308–323, 1985.

[64] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435 (electronic), 1998.

[65] L. I. G. Kovasznay. Laminar flow behind a two-dimensional grid. *Proc. Camb. Philos. Soc.*, 44:58–62, 1948.

[66] M.-C. Lai and Z. Li. A remark on jump conditions for the three-dimensional Navier-Stokes equations involving an immersed moving membrane. *Appl. Math. Lett.*, 14(2):149–154, 2001.

[67] L. D. Landau and E. M. Lifshitz. *Fluid mechanics*. Translated from the Russian by J. B. Sykes and W. H. Reid. Course of Theoretical Physics, Vol. 6. Pergamon Press, London, 1959.

[68] Z. Li and S. R. Lubkin. Numerical analysis of interfacial two-dimensional Stokes flow with discontinuous viscosity and variable surface tension. *Internat. J. Numer. Methods Fluids*, 37(5):525–540, 2001.

[69] S. P. Lin and R. D. Reitz. Drop and spray formation from a liquid jet. *Annu. Rev. Fluid Mech.*, 30:85–105, 1998.

[70] P.-L. Lions. *Mathematical topics in fluid mechanics. Vol. 1*, volume 3 of *Oxford Lecture Series in Mathematics and its Applications*. The Clarendon Press Oxford University Press, New York, 1996. Incompressible models, Oxford Science Publications.

[71] C. Liu and N. J. Walkington. Convergence of numerical approximations of the incompressible Navier–Stokes equations with variable density and viscosity. *SIAM Journal on Numerical Analysis*, 45(3):1287–1304, 2007.

[72] N. Lock, M. Jaeger, M. Medale, and R. Occelli. Local mesh adaptation technique for front tracking problems. *Int. J. Numer. Meth. Fluids*, 28:719–736, 1998.

[73] E. Marchandise and J.-F. Remacle. A stabilized finite element method using a discontinuous level set approach for solving two phase incompressible flows. *J. Comput. Phys.*, 219(2):780–800, 2006.

[74] D. J. Mavriplis. An assessment of linear versus nonlinear multigrid methods for unstructured mesh solvers. *Journal of Computational Physics*, 175(1):302–325, 2002.

[75] T. Maxworthy. Experiments on collisions between solitary waves. *Journal of Fluid Mechanics*, 76:177–185, 1976.

[76] W. Mulder, S. Osher, and J. Sethian. Computing interface motion in compressible gas dynamics. *Journal of Computational Physics*, 100(2):209–228, 1992.

[77] S. Muzaferija and M. Peric. Computation of free-surface flows using finite volume method and moving grids. *Numer. Heat Trans., Part B*, 32:369–384, 1997.

[78] C. K. Newman. *Exponential Integrators for the Incompressible Navier-Stokes Equations.* PhD thesis, Virginia Polytechnic Institute and State University, 2003.

[79] K. Ohmori and N. Saito. On the convergence of finite element solutions to the interface problem for the Stokes system. *J. Comput. Appl. Math.*, 198(1):116–128, 2007.

[80] M. A. Olshanskii and A. Reusken. Navier-Stokes equations in rotation form: a robust multigrid solver for the velocity problem. *SIAM J. Sci. Comput.*, 23(5):1683–1706 (electronic), 2002.

[81] S. Osher and R. P. Fedkiw. *Level set methods and dynamic implicit surfaces.* Springer New York, 2003.

[82] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[83] A. Ouazzi and S. Turek. Efficient multigrid and data structures for edge-oriented FEM stabilization. In *Numerical mathematics and advanced applications*, pages 520–527. Springer, Berlin, 2006.

[84] N. Parolini. *Computational fluid dynamics for naval engineering problems.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2004.

[85] N. Parolini and A. Quarteroni. Mathematical models and numerical simulations for the America's Cup. *Comput. Methods Appl. Mech. Engrg.*, 194(9-11):1001–1026, 2005.

[86] A. Prohl. *Projection and Quasi-Compressibility Methods for Solving the Incompressible Navier-Stokes Equations.* B. G. Teubner Stuttgart, 1997.

[87] C. Prud'homme. A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations. *Scientific Programming*, 14(2):81–110, 2006.

[88] C. Prud'homme. Life: Overview of a unified C++ implementation of the finite and spectral element methods in 1D, 2D and 3D. In *Workshop On State-Of-The-Art In Scientific And Parallel Computing*, Lecture Notes in Computer Science. Springer-Verlag, 2006. Accepted.

[89] Z. Qu. *Unsteady open-channel flow over a mobile bed.* PhD thesis, École Polytechnique Fédérale de Lausanne, 2002.

[90] A. Quarteroni, F. Saleri, and A. Veneziani. Factorization methods for the numerical approximation of Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 188:505–526, 2000.

[91] A. Quarteroni and A. Valli. *Numerical approximation of partial differential equations*, volume 23 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1994.

[92] S. Quinodoz. Calculs de benchmark pour un problème bifluide. Semester project, École Polytechnique Fédérale de Lausanne, June 2007.

[93] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. Technical report, Universität Heidelberg, 1996. Preprints SFB 359, Nummer 96-03.

[94] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *roceedings of the National Academy of Sciences USA*, 93(4):1591–1595, 1996.

[95] J. A. Sethian. *Level set methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1996. Evolving interfaces in geometry, fluid mechanics, computer vision, and materials science.

[96] J. A. Sethian and A. Vladimirsky. Fast methods for the eikonal and related Hamilton-Jacobi equations on unstructured meshes. *Proc. Natl. Acad. Sci. USA*, 97(11):5699–5703 (electronic), 2000.

[97] R. Shuttleworth. A comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. Technical report, University of Maryland, College Park, 2006. http://www.cscamm.umd.edu/publications/shuttleworthcopper_CS-06-03.pdf.

[98] A. Smolianski. *Numerical Modeling of Two-Fluid Interfacial Flows*. PhD thesis, University of Jyväskylä, 2001.

[99] M. Spivak. *A comprehensive introduction to differential geometry. Vol. II*. Publish or Perish Inc., Houston, Tex., third edition, 1999.

[100] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comp. Phys.*, 114:146–159, 1994.

[101] M. Tabata. A convergence property of drag and lift coefficients obtained by surface integral computation. In *Recent developments in domain decomposition methods and flow problems (Kyoto, 1996; Anacapri, 1996)*, volume 11 of *GAKUTO Internat. Ser. Math. Sci. Appl.*, pages 269–275. Gakkōtosho, Tokyo, 1998.

[102] M. Tabata and S. Kaizu. Finite element schemes for two-fluids flow problems. MHF Preprint MHF 2005-23, Faculty of Mathematics, Kyushu University, Fukuoka, Japan, 2005.

[103] N. Tanaka. Global existence of two phase nonhomogeneous viscous incompressible fluid flow. *Comm. Partial Differential Equations*, 18(1-2):41–81, 1993.

[104] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Internat. J. Comput. & Fluids*, 1(1):73–100, 1973.

[105] G. I. Taylor. On the decay of vortices in a viscous fluid. *Philosophical Magazine*, 46:671–674, 1923.

[106] T. Tezduyar, S. Aliabadi, and M. Behr. Enhanced-discretization interface-capturing technique (EDICT) for computation of unsteady flows with interfaces. *Comput. Methods Appl. Mech. Engrg.*, 155:235–248, 1998.

[107] T. E. Tezduyar. Finite element methods for flow problems with moving boundaries and interfaces. *Archives of Computational Methods in Engineering*, 8(2):83–130, 2001.

[108] T. E. Tezduyar. Interface-tracking and interface-capturing techniques for computation of moving boundaries and interfaces. In H. A. Mang, F. G. Rammerstorfer, and J. Eberhardsteiner, editors, *Fifth World Congress on Computational Mechanics*, July 2002.

[109] L. Tobiska and R. Verfürth. Analysis of a streamline diffusion finite element method for the Stokes and Navier-Stokes equations. *SIAM J. Numer. Anal.*, 33(1):107–127, 1996.

[110] A.-K. Tornberg. *Interface Tracking Methods with Application to Multiphase Flows*. PhD thesis, Royal Institute of Technology Stockholm, 2000.

[111] A.-K. Tornberg and B. Engquist. A finite element based level-set method for multiphase flow applications. *Comput. Visual. Sci*, 3:93–101, 2000.

[112] K. Tsuchiya and L.-S. Fan. Near-wake structure of a single gas bubble in a two-dimensional liquid-solid fluidized bed: vortex shedding and wake size variation. *Chem. Engrg. Sci.*, 43(5):1167–1181, 1988.

[113] P. B. Vasconcelos and F. D. d'Almeida. Preconditioned iterative methods for coupled discretizations of fluid flow problems. *IMA J. Numer. Anal.*, 18(3):385–397, 1998.

[114] A. Veneziani. Block factorized preconditioners for high-order accurate in time approximation of the Navier-Stokes equations. *Numerical Methods for Partial Differential Equations*, 19(4):487–510, 2002.

[115] M. Wabro. Coupled algebraic multigrid methods for the Oseen problem. *Comput. Vis. Sci.*, 7(3-4):141–151, 2004.

[116] D. C. Wyatt. Development and assessment of a nonlinear wave prediction methodology for surface vessels. *Journal of ship research*, 44:96, 2000.

[117] S. Zeng, C. Vuik, and P. Wesseling. Numerical solution of the incompressible Navier-Stokes equations by Krylov subspace and multigrid methods. *Adv. Comput. Math.*, 4(1-2):27–49, 1995.

Curriculum Vitæ

Christoph Winkelmann was born on April 2nd, 1977 in Sumiswald (BE), Switzerland. He attended primary and secondary school in Wasen (BE). In 1993, he started an apprenticeship as a technical draftsman at *U. Ammann Maschinenfabrik AG* in Langenthal (BE), which he finished in 1997. Afterwards, he enroled in the *University of Applied Sciencies Bern* in Burgdorf, where he got the degree of a mechanical engineer in 2001. The same year, he was admitted at *ETH Zürich* from which in 2004, he received a masters degree in computational science and engineering, having done his master's thesis under the supervision of Professor Ralf Hiptmair. Since 2004, he has been working as an assistant of Professor Alfio Quarteroni's in the Chair of Modelling and Scientific Computing at *École Polytechnique Fédérale de Lausanne*. His research interests have been free surface flow simulations.